UNIVERSIDADE FEDERAL DO PARANÁ

FELIPE KAUAI PEREIRA

UMA PROPOSTA DE UM ALGORITMO DO TIPO *GREY WOLF OPTIMIZER* PARA O PROBLEMA DE TRANSPORTE COM DESIGNAÇÃO E CARREGAMENTO.

CURITIBA

2021

FELIPE KAUAI PEREIRA

UMA PROPOSTA DE UM ALGORITMO DO TIPO *GREY WOLF OPTIMIZER* PARA O PROBLEMA DE TRANSPORTE COM DESIGNAÇÃO E CARREGAMENTO.

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Métodos Numéricos em Engenharia, área de concentração em Programação Matemática, no programa de pós-graduação em Métodos Numéricos em Engenharia, Setores de Tecnologia e Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. José Eduardo Pécora Junior.

Coorientador: Prof. Dr. Leonardo Silva de Lima.

CURITIBA

2021

CATALOGAÇÃO NA FONTE – SIBI/UFPR

P436p Pereira, Felipe Kauai

Uma proposta de um algoritmo do tipo grey wolf optimizer para o problema de transporte com designação e carregamento [recurso eletrônico]/ Felipe Kauai Pereira - Curitiba, 2021.

Dissertação apresentada ao Programa de Pós-graduação em Métodos Numéricos em Engenharia, Setores de Tecnologia e Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. José Eduardo Pécora Junior
Coorientador: Prof. Dr. Leonardo Silva de Lima.

1. Programação linear. 2. Transporte. 3. Logística. I. Pécora Junior, José Eduardo. II. Lima, Leonardo Silva. III. Título. IV. Universidade Federal do Paraná.

CDD 519.72

Bibliotecária: Vilma Machado CRB9/1563

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **FELIPE KAUAI PEREIRA** intitulada: **UMA PROPOSTA DE UM ALGORITMO DO TIPO GREY WOLF OPTIMIZER PARA O PROBLEMA DE TRANSPORTE COM DESIGNAÇÃO E CARREGAMENTO**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 11 de Fevereiro de 2021.

Assinatura Eletrônica
12/02/2021 13:46:08.0
LEONARDO SILVA DE LIMA
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
09/03/2021 17:01:39.0
GUSTAVO VALENTIM LOCH
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
12/02/2021 09:38:21.0
LUCIANA DE SOUZA PESSÔA
Avaliador Externo (PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO)

A minha família.

AGRADECIMENTOS

Uma vez que nos colocamos a trabalhar em um projeto científico dentro de um programa de pós graduação, é bastante natural centralizarmos as tarefas e, de certo modo, negligenciarmos, pelo menos de um ponto de vista das percepções mais imediatas, as influências externas que modelam e estruturam esse caminho. Essa percepção é equivocada. É incomensurável a importância de meditar sobre o papel das instituições e pessoas que nos cercam, apoiam e ajudam nessa trajetória.

Assim sendo, devo meus agradecimentos primeiramente a meus pais, Margareth e Kirk, e meus irmãos, Rafaele e Michael, que não hesitaram em apoiar-me nas escolhas que fiz, e nos momentos difíceis que tive. Estudar é difícil, e o suporte emocional e psicológico que uma família proporciona é fundamental.

Também ressalto a importância que meus queridos amigos tiveram nesse processo: Charles Vinícius, Lucas de Abreu, Flávia Fernanda Flores (3 F's), Guilherme Farias, Nayara Weber.

Sou bastante grato aos meus orientadores, José Eduardo Pécora Junior e Leonardo Silva de Lima, que estimularam e acreditaram nas minhas ideias. Eu não poderia ter desenvolvido os algoritmos que desenvolvi sem que os dois fornecessem as direções para a busca, a escrita e a estrutura do raciocínio otimizador.

Por fim, agradeço ao programa de Pós Graduação em Métodos Numéricos em Engenharia e ao Grupo de Tecnologia Aplicada à Otimização, da Universidade Federal do Paraná, e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por oferecerem os meios para o meu aprendizado e desenvolvimento das pesquisas que realizei.

Pensar é acolher o real tal qual é, na radicalidade última de si próprio.

Arcangelo R. Buzzi, 1975

RESUMO

O problema de transporte com designação e carregamento (ALTP, do inglês *Assignment and Loading Transportation Problem*) foi recentemente introduzido na literatura inspirado por um problema envolvendo operações militares. O ALTP possui potenciais aplicações em logística de emergência e, como tal, requer algoritmos que constroem soluções aceitáveis em curtos intervalos de tempo. No presente trabalho, um novo algoritmo metaheurístico é proposto para o ALTP. O algoritmo proposto tem por base uma variação da metaheurística *Grey Wolf Optimizer*, que é inspirada na hierarquia social e comportamento de caça de alcateias de lobos cinzas na natureza. O algoritmo apresentado é avaliado em 80 instâncias, compreendendo tanto instâncias reais como artificiais, e comparado com o algoritmo heurístico disponível na literatura para a construção de soluções rápidas para o problema. O desempenho de ambos os algoritmos é avaliado em diferentes limites de tempo para execução. Os resultados experimentais indicam que o algoritmo desenvolvido no presente trabalho é bastante competitivo e pode ser aplicado com sucesso ao novo problema introduzido na literatura. Particularmente, instâncias realísticas são resolvidas com maior eficácia pelo algoritmo desenvolvido, dentro de limites de tempo distintos, quando comparado à heurística disponível na literatura.

Palavras-chave: Problema de transporte com designação e carregamento. Grey Wolf Optimizer. Meta-heurísticas. Otimização. Programação linear inteira mista. Logística de emergência.

ABSTRACT

The Assignment and Loading Transportation Problem (ALTP) was recently introduced in the literature inspired by a problem emerging in military operations. The ALTP has potential applications in emergency logistics and, as such, requires algorithms that provide reasonable solutions in acceptable time limits. In the present work, a new metaheuristic algorithm is proposed for the ALTP. The algorithm developed is based on a recent version of the Grey Wolf Optimizer (GWO), which is inspired in the hierarchical structure and hunting behavior of grey wolves in nature. The algorithm is evaluated in 80 benchmark instances, comprising both realistic and artificial datasets, and compared to the state-of-the-art meta-heuristic in the literature. The performance of both meta-heuristics is studied in different time limits for the execution of the algorithms. Experimental results indicate that the algorithm developed in the present study is quite competitive and can be successfully applied to the newly introduced problem in the literature. Particularly, realistic instances are solved with greater efficacy by the GWO, within distinct time limits, when compared to the algorithm available in the literature.

Keywords: Assignment and loading transportation problem. Grey Wolf Optimizer. Metaheuristics. Optimization. Mixed Integer Linear Programming.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Recentemente Homsi et al. (2019) introduziram na literatura o Problema de Transporte com Designação e Carregamento (ALTP, do inglês *Assignment and Loading Transportation Problem*). O problema é uma generalização do Problema de Mochilas Múltiplas com Designação (MKAP, do inglês *Multiple Knapsack Assignment Problem*), estudado por Kataoka e Yamada (2014), e embora tenha sido concebido para otimização de operações militares do exército dos Estados Unidos da América, sua estrutura tem potencial aplicabilidade em problemas de logística de emergência.

O ALTP é enunciado da seguinte maneira: uma unidade militar requer uma série de itens de diferentes tipos como, por exemplo, alimentos, equipamentos médicos e munição, os quais estão disponíveis em diferentes bases instaladas e devem ser transportados por um conjunto heterogêneo de veículos com diferentes capacidades de transporte. A unidade requer ainda uma quantidade mínima e máxima de cada tipo dos itens, os quais têm um certo valor intrínseco para a unidade, chamado lucro. Cada veículo deve ser designado para no máximo uma base, e existe um custo associado ao deslocamento do veículo para a base, seu carregamento, e o transporte dos itens até a unidade. O problema, então, consiste em encontrar uma partição factível de itens dentre os veículos de maneira que as demandas da unidade sejam satisfeitas, e o lucro geral maximizado.

A unidade militar do problema enunciado anteriormente pode ser substituída por uma base humanitária, ou mesmo por um acampamento montado para operações de socorro em desastres naturais. As aplicabilidades do ALTP, portanto, surgem no contexto das operações de emergência, as quais envolvem complexos problemas na intersecção da pesquisa operacional com a ciência da computação. Por exemplo, resgaste de vítimas e alocação de recursos para áreas afetadas por desastres naturais envolvem problemas de otimização combinatória sofisticados, os quais usualmente incluem transporte, designação e sequenciamento de operações (LINET et al. 2004; DIMITROV et al. 2015; ZHENG et al. 2015). Nesses contextos, requerem-se soluções razoáveis em intervalos de tempo restritos, e algoritmos metaheurísticos, em oposição a métodos exatos, têm um papel fundamental na resolução desses problemas. Porém, pelo problema possuir alta complexidade e, assim, não ser resolvido adequadamente por meio de algoritmos exatos, a necessidade da utilização de heurísticas torna-se imprescindível.

Metaheurísticas são algoritmos aplicados com bastante sucesso em uma extensa gama de problemas complexos como, por exemplo, roteirização de veículos (KORAYEM et al. 2015; BRANDSTÄTTER 2019), e muitos outros (MIRJALILI et al. 2020; QU et al. 2020). Especificamente, alguns desses algoritmos implementam estratégias que simulam o comportamento de populações de espécies de animais como abelhas, formigas e pássaros, para explorar um espaço de soluções em que algoritmos exatos seriam computacionalmente ineficientes (GENDREAU e POTVIN, 2010). Por exemplo, Mirjalili et al. (2014) construíram um algoritmo com base na hierarquia social e estratégia de caça de populações de lobos cinzas para resolução de problemas de variáveis contínuas em engenharia. O algoritmo, conhecido como *Grey Wolf Optimizer* (GWO), destaca-se pela hierarquia de liderança dos agentes dentro do espaço de solução, o que o torna bastante eficiente para resolução de problemas quando comparado a algoritmos concorrentes (FARIS et al. 2018; OSZOYDAN 2019).

No presente trabalho o ALTP é resolvido por meio de uma versão estendida do GWO para o caso onde variáveis inteiras e binárias devem ser determinadas. O novo algoritmo, chamado de agora em diante de ALTP-GWO, é testado em instâncias realísticas e artificiais, e seu desempenho é avaliado em função de variações estruturais nas instâncias estudadas.

1.1 OBJETIVOS DO TRABALHO

1.1.1 Objetivo geral

- Propor uma modificação do algoritmo GWO para o problema ALTP de Programação Linear Inteira Mista.
- Avaliar o desempenho do algoritmo proposto em instâncias realísticas e artificiais, e comparar com a heurística proposta por Homsi et al. (2019).

1.1.2 Objetivos específicos

- Inserir estratégias para determinação de variáveis binárias e inteiras dentro do GWO.

- Avaliar o desempenho do algoritmo em limites de tempo distintos e em função de variações estruturais das instâncias do ALTP.

1.2 ESTRUTURA DO TRABALHO

A apresentação deste estudo está organizada da seguinte maneira. Na Seção 2 a descrição matemática detalhada do ALTP é fornecida. Nesta seção são apresentados os conjuntos, índices e variáveis que serão utilizados no desenvolvimento posterior do trabalho, assim como a estrutura fundamental do problema.

A Seção 3 comporta uma descrição detalhada do GWO na literatura. As principais componentes do algoritmo são descritas. Também são apresentadas algumas aplicações recentes do algoritmo, a fim de fornecer uma melhor contextualização da atual utilização do procedimento na literatura especializada.

A Seção 4 apresenta o desenvolvimento do GWO para o ALTP. Cada componente do algoritmo é desenvolvida em detalhe, onde cada Subseção trata de uma sub-rotina principal do delineamento geral do algoritmo proposto.

Na Seção 5 são descritas as instâncias utilizadas, suas estruturas, assim como o ambiente computacional no qual os experimentos foram executados. Também, os resultados são apresentados.

A Seção 6 compreende uma discussão sobre o comportamento dos algoritmos estudados, assim como potenciais fragilidades encontradas.

Por fim, na Seção 7 são apresentadas as conclusões do trabalho, e potenciais direções de pesquisas futuras.

2 DESCRIÇÃO MATEMÁTICA DO PROBLEMA

No ALTP busca-se transportar um conjunto de itens de diferentes tipos $T = \{1, 2, \dots, t\}$ de um conjunto de bases $B = \{1, 2, \dots, b\}$ (e.g. humanitárias) para uma unidade requerente. A unidade requer no mínimo l_j e no máximo u_j itens de cada tipo $j \in T$, que possui um lucro positivo p_j (valor intrínseco do item para a unidade) e um peso w_j. Para o transporte, considere um conjunto $V = \{1, 2, \dots, v\}$ de diferentes veículos (aviões, ambulâncias, caminhões...) com diferentes capacidades c_i $(i \in V)$. Cada veículo deve ser designado para no máximo uma base k, que possui uma disponibilidade a_{kj} do item do tipo j. O custo de designação do veículo i para a base k e posterior transporte dos itens para a unidade é dado por q_{ik}. O problema, então, consiste em encontrar uma partição factível de itens de cada tipo j dentre os veículos, de tal maneira que as demandas da unidade sejam completamente satisfeitas, e o custo geral minimizado. Na Tabela 1 são apresentados os conjuntos, índices e parâmetros de maneira resumida.

TABELA 1 – SUMÁRIO COM AS NOTAÇÕES USADAS E RESPECTIVAS DESCRIÇÕES

Conjuntos	**Descrição**
T	Conjunto de itens
V	Conjunto de veículos disponíveis para transporte
B	Conjunto de bases
Índices	**Descrição**
j	Tipo do item: $1, 2, \dots, t$
i	Veículo: $1, 2, \dots, v$
k	Base: $1, 2, \dots, b$
Parâmetros	**Descrição**
l_j	Número mínimo de itens do tipo j requeridos pela unidade
u_j	Número máximo de itens do tipo j requeridos pela unidade
p_j	Lucro (valor intrínseco) do item do tipo j para a unidade
w_j	Peso do item do tipo j
c_i	Capacidade do veículo i para transporte
q_{ik}	Custo de deslocamento do veículo i para a base k e para a unidade
a_{kj}	Disponibilidade de itens do tipo j na base k

Seja x_{ikj} uma variável do tipo inteira que representa o número de itens do tipo j recolhidos da base k pelo veículo i e y_{ik} uma variável binária, que assume o valor 1 se o veículo i é designado para a base k , e 0 caso contrário. O problema de Programação Linear Inteira Mista pode ser escrito, então, da seguinte maneira:

$$\max f(x,y) = \sum_{i\in V}\sum_{k\in B}\sum_{j\in T} p_j\, x_{ikj} - \sum_{i\in V}\sum_{k\in B} q_{ik} y_{ik} \quad (1)$$

Sujeito a:

$$\sum_{k\in B} y_{ik} \leq 1 \qquad \forall\, i \in V \quad (2)$$

$$\sum_{i\in V} x_{ikj} \leq a_{kj} \qquad \forall\, k \in B, j \in T \quad (3)$$

$$\sum_{j\in T} w_j x_{ikj} \leq c_i y_{ik} \qquad \forall\, i \in V, k \in B \quad (4)$$

$$l_j \leq \sum_{i\in V}\sum_{k\in B} x_{ikj} \leq u_j \qquad \forall\, j \in T \quad (5)$$

$$x_{ikj} \in \mathbb{Z}^+ \qquad \forall\, i \in V, k \in B, j \in T \quad (6)$$

$$y_{ik} \in \{0,1\} \qquad \forall\, i \in V, k \in B \quad (7)$$

A função objetivo (1) calcula a diferença entre o lucro total e o custo de transporte. As restrições (2) determinam que cada veículo deve ser designado para no máximo uma base. As restrições (3) impõem que o número de itens do tipo j recolhidos da base k não pode exceder a disponibilidade do item j na base k. As restrições (4) impedem que a capacidade dos veículos seja excedida. As restrições (5) estabelecem que o número de itens transportados, de cada tipo, deve respeitar a quantidade mínima de itens requeridos pela unidade, e não deve exceder a quantidade máxima. Finalmente, as restrições (6) e (7) definem o domínio das variáveis x_{ikj} e y_{ik}, onde $\mathbb{Z}^+$ representa o conjunto de inteiros não-negativos.

De maneira a ilustrar um exemplo de solução para o ALTP, consideramos os seguintes dados hipotéticos apresentados na Tabela 2, a seguir. A primeira coluna representa os parâmetros do ALTP, a segunda coluna os valores hipotéticos para o exemplo, e a última coluna apresenta uma descrição do parâmetro correspondente. A solução apresentada, para o conjunto de dados proposto, será construída em um exemplo de execução de uma das sub-rotinas propostas na subseção 4.2. O objetivo neste exemplo, no entanto, é apresentar a estrutura da solução, assim como o valor da sua função objetivo, com base em dados gerados artificialmente.

TABELA 2 – DADOS HIPOTÉTICOS PARA O EXEMPLO DE EXECUÇÃO DO ALGORITMO 2.

Parâmetro	Valor	Descrição
l_1	4	Número mínimo de itens do tipo 1 requerido
l_2	12	Número mínimo de itens do tipo 2 requerido
u_1	12	Número máximo de itens do tipo 1 requerido
u_2	15	Número máximo de itens do tipo 2 requerido
p_1	30	Lucro (valor intrínseco) do item do tipo 1
p_2	40	Lucro (valor intrínseco) do item do tipo 2
w_1	4	Peso do item do tipo 1
w_2	5	Peso do item do tipo 2
c_1	42	Capacidade do veículo 1
c_2	46	Capacidade do veículo 2
q_{11}	100	Custo de designação do veículo 1 para a base 1 e transporte para a unidade
q_{12}	60	Custo de designação do veículo 1 para a base 2 e transporte para a unidade
q_{21}	50	Custo de designação do veículo 2 para a base 1 e transporte para a unidade
q_{22}	130	Custo de designação do veículo 2 para a base 2 e transporte para a unidade
a_{11}	10	Disponibilidade de itens do tipo 1 na base 1
a_{12}	8	Disponibilidade de itens do tipo 2 na base 1
a_{21}	12	Disponibilidade de itens do tipo 1 na base 2
a_{22}	14	Disponibilidade de itens do tipo 2 na base 2

FONTE: O autor (2020)

Uma solução para os dados apresentados na Tabela 2 pode ter a seguinte forma:

$$(\bar{x}, \bar{y}) =: \{\bar{\mathrm{x}}^s_{111} = 0;\ \bar{\mathrm{x}}^s_{121} = 3;\ \bar{\mathrm{x}}^s_{112} = 0;\ \bar{\mathrm{x}}^s_{122} = 6;\ \bar{\mathrm{x}}^s_{211} = 1;\ \bar{\mathrm{x}}^s_{221} = 0;$$
$$\bar{\mathrm{x}}^s_{212} = 8;\ \bar{\mathrm{x}}^s_{222} = 0; \bar{\mathrm{y}}^s_{11} = 0;\ \bar{\mathrm{y}}^s_{12} = 1; \bar{\mathrm{y}}^s_{22} = 0;\ \bar{\mathrm{y}}^s_{21} = 1\}$$

A função objetivo para esta solução terá de acordo com a Equação 1, portanto, o valor $f(\bar{x}, \bar{y}) = 570$.

O ALTP é uma generalização do Problema de Mochilas Múltiplas com Designação (MKAP), estudado por Kataoka e Yamada (2014), com a adição de custos de transporte, e restrições sobre um número mínimo e máximo de itens a serem transportados. Como o MKAP é, por sua vez, uma generalização do Problema de Mochilas Múltiplas, o qual é fortemente $\mathcal{NP}$-hard (MARTELLO e TOTH, 1990), segue-se que o ALTP é também fortemente $\mathcal{NP}$-hard (HOMSI et al., 2019). Ainda, devido às restrições de número mínimo e máximo de itens a serem transportados, algumas instâncias podem não possuir soluções factíveis.

Homsi et al. (2019) propuseram dois métodos para a resolução do ALTP: uma heurística (H-Poly) e uma *Math-heurística*. O algoritmo H-Poly fora desenvolvido para retornar soluções dentro de limites de tempo aceitáveis, uma vez que o problema está

inserido em um contexto de logística de emergência. A *Math-heurística*, por outro lado, requer a utilização de um *solver*, maior poder computacional e maior disponibilidade de tempo para convergência do algoritmo.

O algoritmo heurístico H-Poly opera por meio da construção de uma solução inicial com uma heurística construtiva. Após a obtenção de uma solução inicial, factível ou não, o algoritmo implementa uma busca local iterada (ILS, do inglês *Iterated Local Search*). O ILS realiza uma busca por meio da relaxação das restrições de disponibilidade de itens nas bases, capacidades dos veículos e limites inferior e superior das demandas de cada tipo de item pela unidade. Ao longo das iterações, o algoritmo penaliza a função objetivo de soluções cujas restrições mencionadas anteriormente são violadas. A intensidade das penalizações depende de quais restrições são violadas. Além disso, perturbações são realizadas nas designações de veículos para as bases e na quantidade de itens carregados em cada veículo, de maneira a diversificar as soluções e escapar de potenciais ótimos locais. O algoritmo desenvolvido nesse trabalho será comparado diretamente com H-Poly.

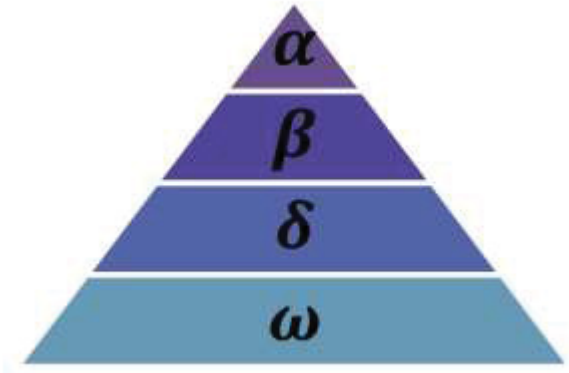
3 REVISÃO DA LITERATURA

3.1 GREY WOLF OPTIMIZER

O algoritmo *Grey Wolf Optimizer* (GWO) foi introduzido na literatura por Mirjalili et al. (2014), e é inspirado na estrutura social de alcateias de lobos cinzas na natureza. O algoritmo simula o comportamento social dos lobos na busca por presas, cercamento e ataque. A liderança da alcateia obedece a uma hierarquia bastante específica, a qual está na estrutura central do algoritmo.

A alcateia possui quatro níveis hierárquicos distintos: o lobo alfa α, lobo beta β, lobo delta δ e, por fim, os lobos ômega Ω. O α está no topo da hierarquia e é responsável pelas decisões do grupo. No segundo nível da pirâmide hierárquica está o β, o qual é responsável por reforçar as decisões de α para os demais lobos da alcateia. No terceiro nível está a sentinela δ, a qual se submete aos lobos α e β, mas domina o resto da alcateia Ω. Os lobos Ω, na base de pirâmide hierárquica, têm o papel de bode expiatório, isto é, realizam buscas para além da localização estimada da presa, e são os indivíduos de menor expressividade no ordenamento do grupo. A Figura 1 demonstra a estrutura hierárquica descrita.

FIGURA 1 – HIERARQUIA SOCIAL DE UMA ALCATEIA. DOMINÂNCIA DECRESCE DE CIMA PARA BAIXO.



FONTE: Adaptado de Mirjalili et al. (2014).

Na simulação do processo de caça e cercamento de presas, o algoritmo GWO utiliza as três melhores soluções, i.e., os três lobos mais bem posicionados no espaço. Denotando as três melhores soluções como X_α, X_β, e X_δ, correspondentes aos lobos α, β e δ, respectivamente, e, utilizando a notação de Luo (2019), as posições dos lobos são atualizadas por meio das seguintes fórmulas iterativas:

$$S1_i^j = X_\alpha^j - a1 * |c1 * X_\alpha^j - X_i^j(t)| \quad (8)$$

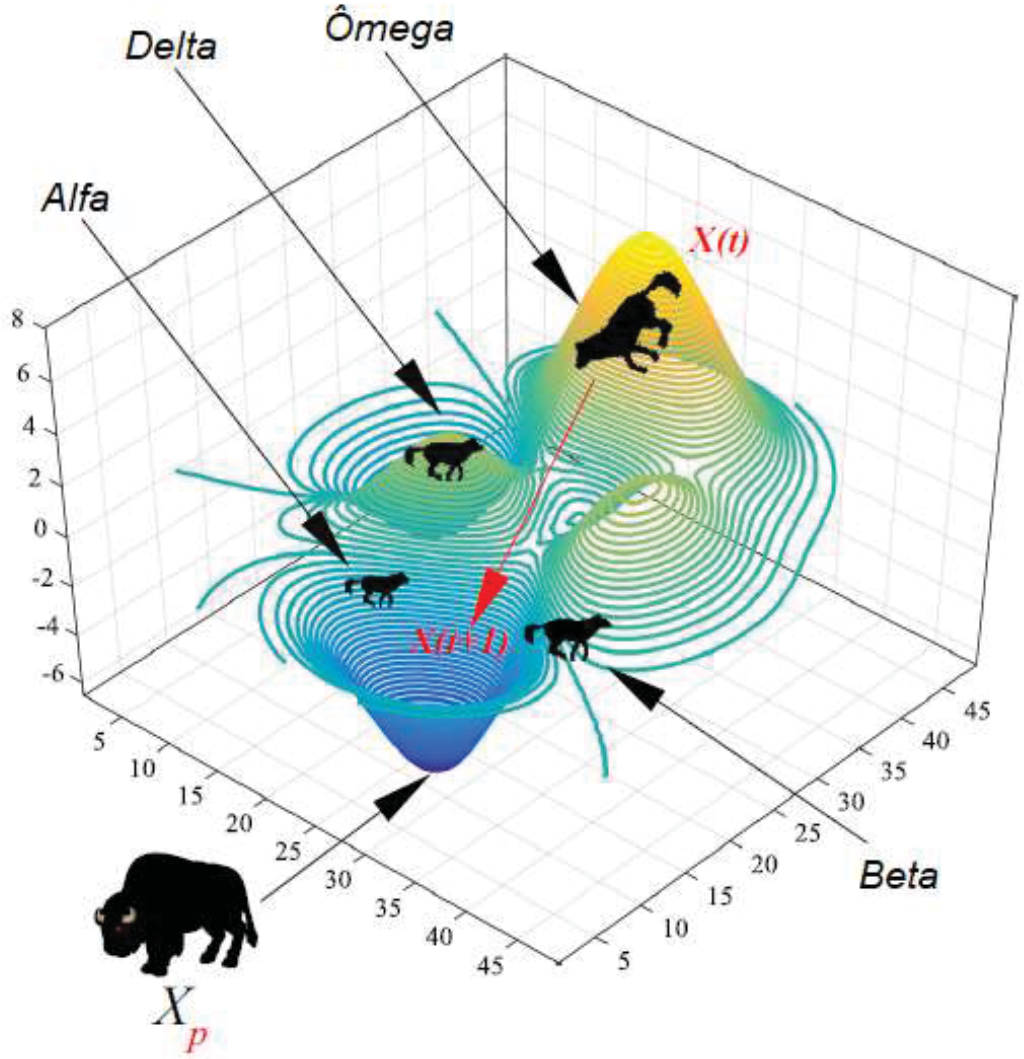
$$S2_i^j = X_\beta^j - a2 * |c2 * X_\beta^j - X_i^j(t)| \quad (9)$$

$$S3_i^j = X_\delta^j - a3 * |c3 * X_\delta^j - X_i^j(t)| \quad (10)$$

$$X_i^j(t+1) = h(S1_i^j, S2_i^j, S3_i^j) \quad (11)$$

onde $X_i^j(t)$ representa o valor da i-ésima solução na j-ésima dimensão da t-ésima iteração. A função h representa a média aritmética, isto é, $(S1_i^j + S2_i^j + S3_i^j)/3$. Os números reais $a1, a2$ e $a3$ são gerados aleatoriamente seguindo a distribuição uniforme no intervalo $[-2 * \left(1 - \frac{t}{G}\right), 2 * \left(1 - \frac{t}{G}\right)]$ e $c1, c2$, e $c3$ são aleatoriamente gerados com distribuição uniforme no intervalo $[0,2]$. G é o número máximo de iterações do algoritmo. Se $|a1| < 1$, $|a2| < 1$ ou $|a3| < 1$, o algoritmo irá explorar soluções contidas em uma vizinhança do ótimo local, caso contrário, o algoritmo busca por soluções mais diversas. Como Mirjalili et al. (2014) argumentam, a compressão progressiva do intervalo para geração dos números reais $a1, a2$ e $a3$, busca modelar a precisão da alcateia na localização da presa à medida que o espaço de busca é mais amplamente percorrido ao longo das iterações. A Figura 2 representa a estrutura hierárquica do GWO na otimização de uma função hipotética.

FIGURA 2 – DEFINIÇÃO DE ALFA, BETA, DELTA E ÔMEGA NO GWO.



FONTE: Adaptado de Faris et al. (2018).

As variáveis $a1, a2$, e $a3$ decrescem linearmente com $t \to G$. Note que se $t > \frac{G}{2}$, $|a1| < 1$, $|a2| < 1$ e $|a3| < 1$. Portanto, na segunda metade do total de iterações, o algoritmo irá produzir soluções ao redor de um ótimo local continuamente. Ademais, o GWO é bastante eficiente em resolver problemas onde o ótimo global encontra-se na origem do sistema de coordenadas (LUO, 2019), porém opera com menor eficácia quando há um desvio, mesmo que pequeno, da origem. Esse problema é resolvido em Luo (2019), por meio de uma versão melhorada do GWO.

A versão melhorada do GWO é construída com base em estimativas dinâmicas sobre a localização da presa, da seguinte maneira:

$$x_j^p(t) = w_\alpha * x_j^\alpha(t) + w_\beta * x_j^\beta(t) + w_\delta * x_j^\delta(t) + \epsilon(t), \quad (12)$$

onde $x_j^p(t)$ representa j-ésima dimensão da localização da presa p na t-ésima iteração. Da mesma maneira, os símbolos α, β e δ referem-se às soluções dos lobos alfa, beta e delta, respectivamente. Os pesos w_α, w_β e w_δ simulam as dominâncias de cada lobo no processo iterativo. De acordo com a hierarquia social da alcateia: $0 \leq w_\delta < w_\beta < w_\alpha \leq 1$. Os pesos são calculados como segue:

$$(w_\alpha, w_\beta, w_\delta) = \left(\frac{f(x^\alpha)}{f(x^\alpha)+f(x^\beta)+f(x^\delta)}, \frac{f(x^\beta)}{f(x^\alpha)+f(x^\beta)+f(x^\delta)}, \frac{f(x^\delta)}{f(x^\alpha)+f(x^\beta)+f(x^\delta)}\right), \quad (13)$$

onde $f(x)$ é o *fitness* da solução x.

O erro estocástico simulado $\epsilon(\mathrm{t}) \sim \mathrm{N}(0, \sigma(\mathrm{t}))$, que segue uma distribuição normal com média zero e desvio padrão $\sigma(t)$, é usado para simular o erro de busca pelos lobos líderes na localização da presa, e o desvio dinâmico $\sigma(t) > \sigma(t+1)$ simula que maior precisão é alcançada no processo de caça na medida em que as iterações evoluem. Em Luo (2019) $\sigma(t) = e^{-100*t/G}$, com t a atual iteração e G o número máximo de iterações do algoritmo.

Cada lobo da população irá avançar em direção à presa de acordo com a seguinte equação:

$$x_j^k(t+1) = x_j^p(t) - r * |x_j^p(t) - x_j^k(t)|, \quad (14)$$

com $x_j^k(t)$ representando o valor da j-ésima dimensão na k-ésima solução da t-ésima iteração, e r um número aleatório gerado com distribuição uniforme no intervalo $[-2,2]$. Novos ótimos locais são explorados se $|r| > 1$, caso contrário o algoritmo explora uma solução aos redores da localização estimada da presa. Diferente do GWO original, o alcance de r não decresce linearmente ao longo das iterações e, portanto, a capacidade de exploração do algoritmo se mantém garantida na segunda metade do número de iterações imposto ao algoritmo. O algoritmo geral é bastante sucinto e pode ser descrito como segue:

Inicialização

Estimar a posição da presa

Construir solução com atualização das posições dos lobos

Calcular *fitness* da solução gerada

Atualizar alfa, beta e delta

repetir

Para melhor esclarecimento da terminologia adotada pelo GWO, a seguir destacamos os principais termos do algoritmo e seus correspondentes na descrição das soluções em problemas de otimização.

- Lobo alfa: melhor solução encontrada para o problema
- Lobo beta: segunda melhor solução encontrada para o problema
- Lobo delta: terceira melhor solução encontrada para o problema
- Lobo ômega: qualquer solução no conjunto de soluções considerado
- Presa: potencial solução para o problema, superior às soluções encontradas anteriormente
- Alcateia: conjunto de soluções para o problema em questão

3.2 APLICAÇÕES DO GWO NA LITERATURA

O GWO foi introduzido na literatura como um algoritmo para a resolução de problemas de otimização globais, e problemas de *design* mecânico. Porém, a estrutura hierárquica do algoritmo, implementada por meio dos lobos dominantes, torna o algoritmo bastante eficiente para diversos problemas de otimização combinatória (OZSOYDAN, 2019). Aplicações recentes, como demonstram Faris et al. (2018), são particularmente concentradas em problemas de engenharia e aprendizagem de máquina.

Variantes do GWO têm sido desenvolvidas para a resolução de diversos problemas de otimização combinatória em engenharia (GUPTA e DEEP, 2019). Por exemplo, aplicações podem ser verificadas em problemas de *Flowshop* (KOMAKI e KAYVANFAR, 2015), seleção de atributos em problemas de classificação (ABDEL-BASSET et al., 2020), planejamento de rota para veículos aéreos não tripulados (ZHANG et al., 2016; DEWANGAN et al., 2019) e problemas de roteirização de veículos capacitados (KORAYEM et al., 2015). Também, a construção de variantes do GWO tem sido realizada para problemas clássicos, como o Problema do Caixeiro Viajante (WANG et al., 2020).

Recentemente, Luo e Zhao (2019) desenvolveram uma variante do GWO para o Problema das Mochilas Multidimensional (MKP, do inglês *Multidimensional Knapsack Problem*) e, embora o algoritmo fora desenvolvido especificamente para o MKP, seu delineamento geral pode ser aplicado a qualquer problema de otimização

combinatória. Os autores introduzem uma função que transforma soluções contínuas, recuperadas pelo GWO, em soluções binárias e comparam o novo algoritmo com os três algoritmos mais competitivos da literatura recente para o MKP; *Hybrid harmony search-based Algorithm* (ZHANG et al., 2015), *Binary fruit fly Algorithm* (WANG et al., 2013) e *Hybrid quantum inspired harmony search Algorithm* (LAYEB, 2013). A versão binária desenvolvida pelos autores demonstrou superioridade em dois conjuntos de instâncias bastante conhecidos na literatura para o problema.

Em aprendizagem de máquina, hibridizações e variantes do algoritmo têm sido estudadas em redes neurais artificiais multicamadas, na determinação dos pesos e vieses (*bias*) ótimos para um conjunto de amostras de treinamento, demonstrando alta competitividade em relação a metaheurísticas de diferentes estruturas (MIRJALILI, 2015, RASHID et al., 2019). Além disso, diversos trabalhos têm utilizado o GWO para otimização dos parâmetros de Máquinas de Vetores de Suporte (BIAN et al., 2017; SAXENA e SHEKHAWAT, 2017; ROOPA DEVI e SUGANTHE, 2020), assim como para problemas de análise de agrupamento (KAPOOR et al., 2017; TRIPATHI et al., 2018).

Em trabalho recente, Tripathi et al. (2018) introduzem uma variante do GWO para o problema de análise de agrupamento. O algoritmo, chamado de EGWO (do inglês, *Enhanced Grey Wolf Optimizer*), apresentou maior eficiência no agrupamento de dados que algoritmos comumente utilizados para esse propósito, como o *Particle Swarm Optimization* (PSO), *Gravitational Search Algorithm* (GSA), *Bat Algorithm* (BA) e o K-médias. Além disso, os autores paralelizam o algoritmo por meio do estilo arquitetural MapReduce e, assim, obtém um algoritmo ainda mais robusto que o próprio EGWO.

Faris et al. (2019) modelam um problema de otimização combinatória para encontrar o número de neurônios em camadas intermediárias e os pesos das conexões em Redes Neurais Artificiais (ANN, do inglês *Artificial Neural Networks*), de maneira ótima. Os autores implementam uma série de meta-heurísticas com base em populações para o problema introduzido, e verificam que dentre todas as meta-heurísticas avaliadas, o GWO obteve o melhor desempenho. Embora nenhuma mudança significativa na estrutura do GWO original tenha sido aplicada, os autores destacam potenciais nichos de melhoria do algoritmo para o problema proposto.

Devido ao bom desempenho apresentado pelo GWO em uma série de problemas de otimização combinatória, faz-se necessário verificar se o algoritmo pode

ser aplicado com sucesso também ao ALTP. O presente trabalho é o primeiro a propor uma variante do GWO para o ALTP. Aqui são desenvolvidos mecanismos para a construção de soluções onde variáveis inteiras não-negativas e binárias compõem o problema. Portanto, também, as tecnologias desenvolvidas no algoritmo proposto podem ser utilizadas em problemas distintos de Programação Linear Inteira e Mista.

4 DESENVOLVIMENTO DO GWO PARA O ALTP

4.1 DELINEAMENTO GERAL DO ALTP-GWO

Uma solução para o ALTP possui uma matriz de números inteiros $\bar{\boldsymbol{x}}$ e uma matriz com entradas binárias $\bar{\boldsymbol{y}}$. A matriz $\bar{\boldsymbol{x}}$ representa o número de itens do tipo j carregados da base k pelo veículo i, e seus elementos são da forma $\bar{x}_{ikj}$. A matriz binária $\bar{\boldsymbol{y}}$ representa a designação do veículo i para a base k, e seus elementos possuem a forma $\bar{y}_{ik}$, onde $\bar{y}_{ik} = 1$ se o veículo i é designado para a base k, e $\bar{y}_{ik} = 0$ caso contrário. Para simplificação de notação, denotamos a $n\text{-}ésima$ solução como $(\bar{x}^n, \bar{y}^n)$.

O algoritmo ALTP-GWO é consideravelmente conciso e segue uma estrutura muito semelhante ao algoritmo introduzido por Luo e Zhao (2019) com estrutura binária. O delineamento geral do algoritmo é apresentado no Algoritmo 1. A inicialização do algoritmo ocorre com a geração da população inicial de soluções de tamanho *S*, por meio da função *GenerateElitePop*(*S*), e a identificação dos lobos líderes α, β e δ. A execução do algoritmo é controlada pelo iterador t, e possui como critério de parada um número máximo de iterações denotado por G, conforme a Linha 4.

Após a inicialização existem dois laços de repetição. O laço externo controla o número máximo de iterações do algoritmo. O laço interno, representado pelas instruções contidas entre a Linhas 6 e 28, implementa as mudanças de posições para cada lobo da alcateia e penaliza a função objetivo caso a solução retornada seja infactível. A penalização é calculada como $f(\bar{x}, \bar{y})(1 - \Delta)$, onde Δ é um parâmetro do algoritmo que representa a intensidade da penalização, cada vez que o número de itens de cada tipo j transportado viola sua respectiva restrição *lower bound*, isto é, o número mínimo de itens de um tipo j requerido pela unidade. Note que a factibilidade da solução é comprometida apenas pelas restrições (5) no ALTP.

O ALTP-GWO possui três sub-rotinas principais. A primeira sub-rotina concerne o gerador da população inicial, $GenerateElitePop()$, que aparece na Linha 1 do Algoritmo 1 e será detalhada na Subseção 4.2. A segunda sub-rotina é responsável pela construção de uma solução experimental, isto é, a partir da estimativa de localização das presas o algoritmo movimenta os lobos, ou soluções, em direção a essa localização. O algoritmo para a construção da solução experimental

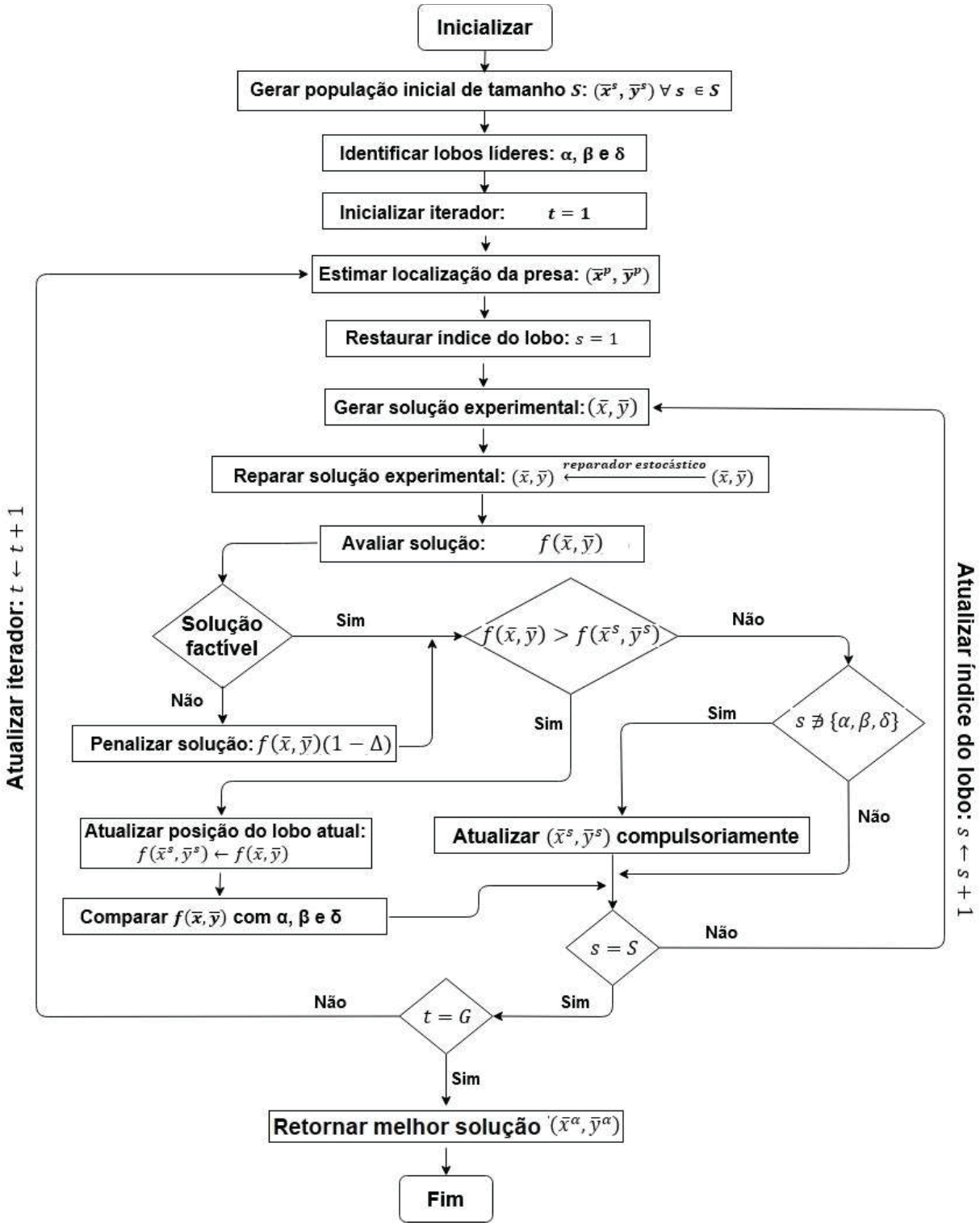
$(\bar{x}, \bar{y})$, denotado pela função *GenerateExpSol*() na Linha 7 do Algoritmo 1, tem como parâmetros a localização estimada da presa $(\bar{x}^p, \bar{y}^p)$ e a solução atual $(\bar{x}^s, \bar{y}^s)$, e será descrito com maiores detalhes na Subseção 4.3. A terceira sub-rotina, representada pela função *ExpSolRepair*() na Linha 8 do Algoritmo 1, aplica reparos na solução experimental obtida anteriormente e será detalhada na Subseção 4.4. Reparos são necessários na medida em que a solução experimental pode retornar soluções infactíveis. Além disso, no ALTP-GWO, a terceira sub-rotina tem a função de adicionar diversidade ao sistema. O delineamento geral do algoritmo proposto é apresentado no Algoritmo 1 e na Figura 3.

Algoritmo 1. *Grey Wolf Optimizer* para o ALTP

```
1: GenerateElitePop(S)
2: Identificar lobos líderes: α, β e δ
3: Inicializar iterador: t ← 1
4: while t ≤ G do
5:     Estimar localização da presas: (x̄^p, ȳ^p)
6:     for s = 1 to S do
7:         (x̄, ȳ) ← GenerateExpSol(x̄^p, ȳ^p, x̄^s, ȳ^s)
8:         (x̄, ȳ) ← ExpSolRepair(x̄, ȳ)
9:         Avaliar solução experimental pela função objetivo na Eq. (1)
10:        Para cada tipo de item que viola sua respectiva restrição
11:        lower bound: f(x̄, ȳ) ← f(x̄, ȳ)(1 − Δ)
12:        if f(x̄, ȳ) > f(x̄^s, ȳ^s) then
13:            (x̄^s, ȳ^s) ← (x̄, ȳ)
14:            Comparar a solução atual com lobos líderes
15:            if f(x̄, ȳ) > f(x̄^α, ȳ^α) then
16:                (x̄^α, ȳ^α) ← (x̄, ȳ)
17:                t ← 1
18:            else if f(x̄, ȳ) > f(x̄^β, ȳ^β) then
19:                (x̄^β, ȳ^β) ← (x̄, ȳ)
20:                t ← 1
21:            else if f(x̄, ȳ) > f(x̄^δ, ȳ^δ) then
22:                (x̄^δ, ȳ^δ) ← (x̄, ȳ)
23:                t ← 1
24:            end
25:        else if o lobo atual não é um líder then
26:            (x̄^s, ȳ^s) ← (x̄, ȳ)
27:        end
28:    end
29:    t ← t + 1
30: end
```

FONTE: O autor (2020)

FIGURA 3 – FLUXOGRAMA DO ALTP-GWO.



FONTE: O autor (2020)

4.2 GERADOR DA POPULAÇÃO INICIAL

Ter uma população inicial com uma *fitness* média de qualidade, assim como boa diversidade, é essencial para dar início ao processo de busca iterativa por soluções sem que o algoritmo fique limitado rapidamente por buscas nas vizinhanças de ótimos locais.

O gerador de população inicial, denotado por *GenerateElitePop*(), opera por meio da inicialização do tamanho S da alcateia (*pack size*), i.e., a quantidade de soluções definida pelo usuário. Na sequência, os veículos são ordenados de maneira não-decrescente com relação aos seus custos de designação q_{ik} por meio da função *SortVehicles*(q_{ik}) na Linha 1 do Algoritmo 2. A ordenação dos veículos é realizada de tal maneira que assim que um veículo é alocado na lista ordenada, esse mesmo veículo não é mais considerado nas avaliações subsequentes de q_{ik}. Assim, o número de veículos ordenados é exatamente o mesmo que o número de veículos disponíveis para transporte. Finalmente, a função *SortItems*() representada na Linha 2 do Algoritmo 2, irá ordenar os itens por razões $\mathrm{p_j}/\mathrm{w_j}$ de maneira não-crescente. Essa razão representa a pseudo-utilidade dos itens, utilizada em ampla escala nos algoritmos desenvolvidos para o Problema das Mochilas Múltiplas na literatura especializada (PUCHINGER et al., 2010; LUO, 2019; HE et al., 2019).

Para cada solução s na alcateia de tamanho S o algoritmo gerador da população inicial inicializa as matrizes $\bar{x}^s$ e $\bar{y}^s$ com soluções vazias, o número de itens do tipo j já carregados o qual denotamos por π_j, a disponibilidade residual de itens do tipo j na base k denotada por θ_{kj}, e o número de itens do tipo j a ser transportado, denotado por μ_j, o qual é aleatoriamente gerado com distribuição uniforme no intervalo $[l_j, u_j]$, de acordo com as Linhas 8, 9 e 10. Definimos *rand(l,u)* uma função que gera um número real aleatório com distribuição uniforme entre l e u, onde $u > l$.

Para cada veículo na lista ordenada de q_{ik}, a capacidade residual φ do veículo recebe sua capacidade total $\mathrm{c_i}$, e cada tipo de item é carregado no veículo de acordo com a ordem não-crescente das razões $\mathrm{p_j}/\mathrm{w_j}$ até que uma de três condições sejam satisfeitas: i) a base para a qual o veículo é designado não possui mais itens disponíveis, ii) a capacidade do veículo é completamente utilizada, iii) o número de itens a serem transportados μ_j foi alcançado. A equação da Linha 14 do Algoritmo 2

representa as três condições citadas. Por fim, se a soma de itens carregados é maior que zero, então o veículo é designado e a iteração continua, de acordo com as Linhas 19 e 20. Note que o número de itens a ser transportado, gerado de maneira aleatória, determina o número de designações realizadas. O pseudo-código do algoritmo gerador da população inicial é apresentado abaixo, no Algoritmo 2, e a seguir um exemplo ilustrativo da execução do algoritmo é apresentado.

Algoritmo 2. *GenerateElitePop*(*S*)

1: $SortVehicles()$
2: $SortItems()$
3: **for** $s = 1$ to S **do**
4: Inicializar $\bar{x}^s$ e $\bar{y}^s$.
5: $\pi_j \leftarrow 0 \quad \forall j \in T$
6: $\theta_{kj} \leftarrow a_{kj} \quad \forall k \in B, j \in T$
7: Inicializar μ_j /* número aleatório de itens de tipo j a serem carregados */
8: **for** $j = 1$ to T **do**
9: $\mu_j \leftarrow \lfloor rand(l_j, u_j) \rfloor$ /* $\lfloor x \rfloor$ é o piso de x */
10: **end**
11: **for** $q = q_1$ to q_m **do** /* em ordem não-decrescente de q_{ik}. Note $q \equiv (i, k)$ */
12: $\varphi \leftarrow c_i$
13: **for** $j = j_1$ to j_n **do** /* em ordem não-crescente de p_j/w_j */
14: $\bar{x}^s_{qj} \leftarrow \max\left\{0, \min\left\{\theta_{kj}, \left\lfloor \frac{\varphi}{w_j} \right\rfloor, \mu_j - \pi_j\right\}\right\}$
15: $\theta_{kj} \leftarrow \theta_{kj} - \bar{x}^s_{qj}$
16: $\varphi \leftarrow \varphi - w_j * \bar{x}^s_{qj}$
17: $\pi_j \leftarrow \pi_j + \bar{x}^s_{qj}$
18: **end**
19: **if** $\sum_{j \in T} \bar{x}^s_{qj} > 0$ **then**
20: $\bar{y}^s_q \leftarrow 1$
21: **end**
22: **end**
23: **end**

FONTE: O autor (2020)

Consideremos como exemplo o primeiro passo da construção de uma solução onde estão disponíveis para transporte 2 veículos, 2 tipos de itens, 2 bases e uma unidade que requer quantidades distintas de itens de cada tipo. Os dados são os mesmos utilizados na Tabela 2.

Inicializar

$S = 1$
$SortVehicles(q_{ik}) = \{q_{21}, q_{12}\}$
$SortItems() = \{2, 1\}$
Inicializar $\bar{x}^1$ e $\bar{y}^1$
$\pi_1 = 0;\ \pi_2 = 0$
$\theta_{11} = 10;\ \theta_{12} = 8;\ \theta_{21} = 12;\ \theta_{22} = 14$
$\mu_1 = 10;\ \mu_2 = 14$ /* gerados aleatoriamente nos intervalos [4, 12] e [12, 15] */

Para o veículo 2 na base 1
$\varphi = 46$

Para o item do tipo 2
$\bar{x}^1_{212} = \max\left\{0, \min\left\{\theta_{12}, \left\lfloor\frac{\varphi}{w_2}\right\rfloor, \mu_2 - \pi_2\right\}\right\} = \max\{0, \min\{8, 9, 14\}\} = 8$
$\theta_{12} = \theta_{12} - \bar{x}^1_{212} = 8 - 8 = 0$
$\varphi = \varphi - w_2 * \bar{x}^1_{212} = 46 - 40 = 6$
$\pi_2 = \pi_2 + \bar{x}^1_{212} = 0 + 8 = 8$

Para o item do tipo 1

$\bar{x}^1_{211} = \max\left\{0, \min\left\{\theta_{11}, \left\lfloor\frac{\varphi}{w_1}\right\rfloor, \mu_1 - \pi_1\right\}\right\} = \max\{0, \min\{10, 1, 10\}\} = 1$
$\theta_{11} = \theta_{11} - \bar{x}^1_{211} = 10 - 1 = 8$
$\varphi = \varphi - w_2 * \bar{x}^1_{212} = 6 - 4 = 2$
$\pi_1 = \pi_1 + \bar{x}^1_{211} = 0 + 1 = 1$
$\sum_{j \in T} \bar{x}^1_{21j} = 9 > 0$
$\bar{y}^s_{21} = 1$

Solução no primeiro passo: $\{\bar{x}^s_{111} = 0;\ \bar{x}^s_{121} = 0;\ \bar{x}^s_{112} = 0;\ \bar{x}^s_{122} = 0;\ \bar{x}^s_{211} = 1;$
$\bar{x}^s_{221} = 0;\ \bar{x}^s_{212} = 8;\ \bar{x}^s_{222} = 0; \bar{y}^s_{11} = 0;\ \bar{y}^s_{12} = 0;$
$\bar{y}^s_{22} = 0;\ \bar{y}^s_{21} = 1\}$

Para o veículo 1 na base 2 ...

Solução final: $\{\bar{x}^s_{111} = 0;\ \bar{x}^s_{121} = 3;\ \bar{x}^s_{112} = 0;\ \bar{x}^s_{122} = 6;\ \bar{x}^s_{211} = 1;\ \bar{x}^s_{221} = 0;$
$\bar{x}^s_{212} = 8;\ \bar{x}^s_{222} = 0; \bar{y}^s_{11} = 0;\ \bar{y}^s_{12} = 1; \bar{y}^s_{22} = 0;\ \bar{y}^s_{21} = 1\}$

fim

4.3 SOLUÇÃO EXPERIMENTAL

O algoritmo para construção da solução experimental, denotado pela função *GenerateExpSol*(), está representado no Algoritmo 3. Os parâmetros de entrada são a localização estimada da presa $(\bar{x}^p, \bar{y}^p)$, e a posição do lobo atual $(\bar{x}^s, \bar{y}^s)$. A localização da presa é estimada de acordo com as Equações (12) e (13). Como para o ALTP existem duas matrizes que compõem a solução, a localização estimada das presas é calculada para as duas variáveis como segue:

$$x_{ikj}^{p}(t) = w_\alpha * x_{ikj}^{\alpha}(t) + w_\beta * x_{ikj}^{\beta}(t) + w_\delta * x_{ikj}^{\delta}(t) \quad (15)$$

$$y_{ik}^{p}(t) = w_\alpha * y_{ik}^{\alpha}(t) + w_\beta * y_{ik}^{\beta}(t) + w_\delta * y_{ik}^{\delta}(t) \quad (16)$$

Note que a localização estimada das presas não retornará soluções necessariamente factíveis, dado que o resultado pertence ao conjunto de número reais não-inteiros. De fato, as estimativas das localizações da presa fornecem apenas uma orientação para o deslocamento dos lobos.

A partir dos parâmetros de entrada, uma solução experimental $(\bar{x}, \bar{y})$ é construída. O procedimento geral consiste em verificar as designações realizadas para o transporte de cada tipo de item. O Algoritmo 3 que descreve a construção da solução experimental possui três laços *for*. O primeiro laço controla o tipo de item sob análise, e o dois laços subsequentes verificam as bases e os veículos designados para cada base, respectivamente.

Designações são realizadas de acordo com o valor de $y_{ik}^{p}(t)$, o qual está contido no intervalo $[0,1]$, onde $y_{ik}^{p}(t) = 0$ se nenhum lobo líder compartilha a posição e 1 se todos os lobos compartilham a mesma posição. Note que $y_{ik}^{p}(t)$ admite números reais não-inteiros devido a equação (16), pois w_α, w_β e w_δ são pesos calculados com base na dominância de cada solução representada pelos lobos de maior hierarquia, de acordo com a Equação (13).

A decisão de designação é realizada por meio da geração de um número aleatório com distribuição uniforme no intervalo $[0,1]$, onde se o valor é menor que $y_{ik}^{p}(t)$, então o veículo atual i é designado para a base k. Esse procedimento é construído com base no seguinte raciocínio: se dois dos lobos de maior hierarquia (alfa, beta e delta) possuem em suas soluções uma designação do veículo i para a

base k (por exemplo, $y_{ik}^{\alpha}(t) = 1$ e $y_{ik}^{\beta}(t) = 1$), então a equação (16) retorna $y_{ik}^{p}(t) > 0.5$, aumentando a chances de designação do veículo i para a base k na solução experimental, de acordo com a Linha 5 do Algoritmo 3.

No caso em que não há designações realizadas pelo procedimento anterior, a posição do lobo atual $\bar{y}_{ik}^{s}(\mathrm{t})$ é verificada, e se a designação é constatada, o algoritmo designa o veículo i para a base k na solução experimental com probabilidade 0.5 (valor obtido com base em experimentos computacionais). Se nenhuma designação é realizada nos passos anteriores, então a designação não ocorre na solução experimental.

Após uma designação realizada, o algoritmo procede com o deslocamento do lobo em direção à posição da presa. Este procedimento consiste em decidir sobre a quantidade de itens a serem transportados na atual designação de acordo com a posição estimada da presa $\bar{x}^{p}$. A construção da solução $\mathrm{t}+1$ é realizada por meio da geração de um número aleatório r com distribuição uniforme no intervalo [-2,2]. Se $|r| > 1$, então:

$$\bar{x}_{ikj}^{s}(\mathrm{t}+1) = x_{ikj}^{p}(t) - r * |\bar{x}_{ikj}^{s}(\mathrm{t}) - x_{ikj}^{p}(t)| \tag{17}$$

caso contrário,

$$\bar{x}_{ikj}^{s}(\mathrm{t}+1) = x_{ikj}^{p}(t) \tag{18}$$

Note que as equações (17) e (18) podem produzir valores não-inteiros. Assim, para que o valor de $\bar{x}_{ikj}^{s}(\mathrm{t}+1)$ seja inteiro, utiliza-se a seguinte regra:

$$\Phi = \bar{x}_{ikj}^{s} - \lfloor \bar{x}_{ikj}^{s} \rfloor \tag{19}$$

$$\bar{x}_{ikj}^{s} = \begin{cases} \lfloor \bar{x}_{ikj}^{s} \rfloor + 1, & \Phi \geq 0.5 \\ \lfloor \bar{x}_{ikj}^{s} \rfloor, & caso\ contrário \end{cases} \tag{20}$$

onde $\lfloor \bar{x}_{ikj}^{s} \rfloor$ é o maior inteiro de $\bar{x}_{ikj}^{s}$ menor que $\bar{x}_{ikj}^{s}$.

O algoritmo controla a quantidade máxima de itens de cada tipo requerida pela unidade, a disponibilidade de itens em cada base e as restrições sobre a capacidade dos veículos, de acordo com as Linhas 9, 15 e 24 do Algoritmo 3. Note que como o algoritmo não considera as restrições *lower bound* da quantidade de itens requerida

pela unidade, soluções infactíveis podem ser construídas. Ainda, designações são realizadas em torno dos mesmos veículos e bases produzidas pelo Algoritmo 2. Assim, na próxima subseção, um operador estocástico para o reparo da solução experimental é apresentado. O pseudo-código do algoritmo da solução experimental é apresentado no Algoritmo 3.

Algoritmo 3. *GenerateExpSol*($\bar{x}^p$, $\bar{y}^p$, $\bar{x}^s$, $\bar{y}^s$)

```
Inicializar solução experimental (x̄, ȳ)
for j = 1 to T do
    for k = 1 to B do
        for i = 1 to V do
            if rand(0,1) < y^p_ik(t) e o veículo i ainda não foi designado then
                r ← rand(−2,2)
                if |r| > 1 then
                    calcular x̄^s_ikj(t + 1) de acordo com (17), (19) e (20)
                    if restrições upper bound são satisfeitas then
                        atualizar quantidade de itens do tipo j já carregados
                        ȳ_ik ← 1;  x̄_ikj ← x̄^s_ikj(t + 1)
                    end
                else
                    calcular x̄^s_ikj(t + 1) de acordo com (18), (19) e (20)
                    if restrições upper bound são satisfeitas then
                        atualizar quantidade de itens do tipo j já carregados
                        ȳ_ik ← 1; x̄_ikj ← x̄^s_ikj(t + 1)
                    end
                end
            else if ȳ^s_ik(t) = 1 e rand(0,1) < 0.5
                        e o veículo i ainda não foi designado
            then
                calcular x̄^s_ikj(t + 1) de acorco com (17), (19) e (20)
                if restrições upper bound são satisfeitas then
                    atualizar quantidade de itens do tipo j já carregados
                    ȳ_ik ← 1; x̄_ikj ← x̄^s_ikj(t + 1)
                end
            end
        end
    end
end
return solução experimental (x̄, ȳ)
```

FONTE: O autor (2020)

4.4 OPERADOR ESTOCÁSTICO DE REPARO DA SOLUÇÃO

O operador estocástico de reparo da solução, denotado pela função *ExpSolRepair*(), recebe como parâmetro a solução experimental atual $(\bar{x}, \bar{y})$. O procedimento inicia com a verificação sobre se existem quaisquer itens que não respeitam as restrições de *lower bound* representadas por l_j. Se qualquer restrição é violada, então o algoritmo executa uma simples busca local, denotada pela função *load*(), que irá tentar carregar um item do tipo não satisfeito por vez para cada designação disponível, respeitando as restrições de disponibilidade nas bases, quantidade máxima de itens requerida e de capacidades dos veículos. Se no curso da execução de *load*(), a restrição violada é restaurada, então a busca cessa, e o próximo item é verificado.

Se depois da execução de *load*() ainda existirem restrições violadas, então o algoritmo encontra uma base com a maior disponibilidade do item para o qual a restrição *lower bound* não é satisfeita, e realiza uma designação de um veículo para a base encontrada com o menor custo possível. Esse procedimento é chamado de *assign*(). Após uma designação realizada, o algoritmo chama *load*() novamente. Esses procedimentos estão representados pelas Linhas de 1 a 11 do Algoritmo 4.

Ainda, mutações são realizadas por meio da troca aleatória de designações realizadas com uma probabilidade ρ. O procedimento é chamado de *suffle*() e funciona da seguinte maneira:

Inicialização

Excluir designação de um veículo aleatório de sua base

Descarregar os seus itens

Designar aleatoriamente um veículo não utilizado para uma base qualquer

Carregar os itens previamente descarregados no novo veículo, respeitando as restrições de capacidade do veículo e disponibilidade do item na base

fim

A probabilidade ρ para chamar a função *shuffle*() não é fixa, mas ajustada dinamicamente ao longo do curso de iterações de acordo com a seguinte fórmula:

$$\rho = \frac{1}{2}(1 - \frac{t}{G}), \tag{21}$$

onde t é a iteração atual e G é o número máximo de iterações definidos para a execução do GWO. O algoritmo inicia com probabilidade $\rho = 0.5$ e procede com o decremento de ρ até $\rho = 0$ quando $t = G$. Esse procedimento, representado pelas Linhas 12, 13 e 14 do Algoritmo 4, tem por objetivo simular uma busca aleatória pela alcateia no início do procedimento, que converge para uma posição definida à medida que a localização da presa se torna melhor conhecida ao longo das iterações.

Note que quando os procedimentos descritos anteriormente terminam, ainda podem existir capacidades residuais para o carregamento de itens. Se esse é o caso, o algoritmo irá carregar um item por vez em uma ordem não-crescente das razões $\mathrm{p_j/w_j}$, de acordo com as Linhas 15-25. O pseudo-código para o operador estocástico de reparo da solução é dado no Algoritmo 4.

Algoritmo 4. *ExpSolRepair*($\bar{x}, \bar{y}$)

```
1:  for j = 1 to T do
2:      if número de itens do tipo j < lower bound l_j then
3:          load();
4:      end
5:  end
6:  for j = 1 to T do
7:      if número de itens do tipo j < lower bound l_j then
8:          assign();
9:          load();
10:     end
11: end
12: if rand(0,1) < ρ then
13:     shuffle();
14: end
15: for i = 1 to V do
16:     for k = 1 to B do
17:         if ȳ_ik(t) = 1 then
18:             for j = j_1 to j_n do /* em ordem não-crescente de p_j/w_j */
19:                 if restrições upper bound são satisfeitas then
20:                     x̄_ikj ← x̄_ikj + 1;
21:                 end
22:             end
23:         end
24:     end
25: end
```

FONTE: O autor (2020)

5 EXPERIMENTOS COMPUTACIONAIS E RESULTADOS

5.1 INSTÂNCIAS PARA TESTE DE MESA

O ALTP-GWO é avaliado em dois grupos de instâncias geradas por Homsi et al. (2019). O primeiro grupo compreende 30 instâncias realísticas, produzidas com base em dados de operações militares obtidos do departamento de defesa dos Estados Unidos da América. O segundo grupo é composto por 50 instâncias geradas aleatoriamente, mas com o mesmo número de tipos de itens, veículos e bases que as instâncias realísticas. Cada instância possui 11 tipos de itens, 8496 veículos, e 12 bases.

As instâncias realísticas são ainda divididas em 6 subgrupos com 5 instâncias cada, e possuem as seguintes propriedades:

- REA-a: a disponibilidade de tipos de itens em cada base é uniformemente distribuída entre 90% e 110% das disponibilidades dos dados militares reais;
- REA-b: restrições *lower* e *upper bound* para cada tipo de item são uniformemente distribuídas entre 90% e 110% das restrições do conjunto de dados reais;
- REA-c: as capacidades dos veículos são uniformemente distribuídas entre 90% e 110% das capacidades do conjunto de dados reais;
- REA-p: os lucros de cada tipo de item são uniformemente distribuídos entre 90% e 110% dos lucros do conjunto de dados reais;
- REA-q: o custo de designação de cada veículo para cada base é uniformemente distribuído entre 90% e 110% dos custos do conjunto de dados reais;
- REA-w: os pesos de cada tipo de item são uniformemente distribuídos entre 90% e 110% dos pesos do conjunto de dados reais.

Todas as instâncias utilizadas estão publicamente disponíveis no repositório do Grupo de Pesquisa Operacional da Universidade de Bologna, no endereço http://or.dei.unibo.it/library/.

5.2 AMBIENTE COMPUTACIONAL E CONFIGURAÇÃO DOS PARÂMETROS

Para avaliar o desempenho do algoritmo proposto, cada instância foi resolvida 30 vezes pelo ALTP-GWO. A principal medida de qualidade adotada foi o *Gap*, o qual foi calculado com base na resolução do problema linear relaxado com o CPLEX 12.6, excluindo-se as restrições de integralidade (6) e (7). O Gap é calculado como segue:

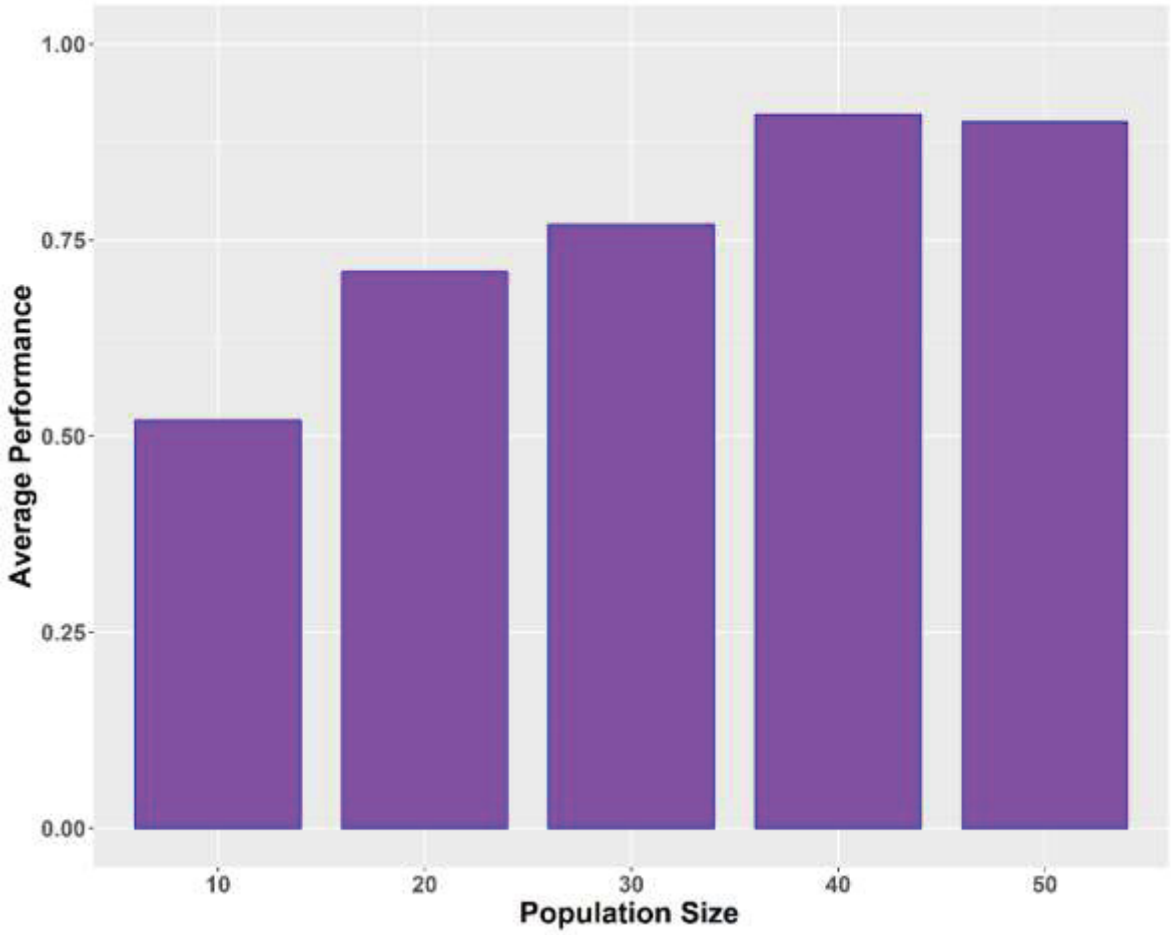
$$Gap = \frac{UB - LB}{UB} * 100 \quad (22)$$

onde UB representa o *upper bound* obtido pela solução do problema linear relaxado e LB representa o *lower bound* obtido pelo ALTP-GWO.

Todas as instâncias foram resolvidas em um Intel® Core™ @2.10 GHz com 64 GB de RAM. O número máximo de iterações sem melhoria G foi configurado em 100. Ainda, o desempenho do ALTP-GWO foi avaliado em diferentes limites de tempo, com 30 execuções independentes sobre as instâncias realísticas Rea-a-1, Rea-b-1, Rea-c-1, Rea-q-1, Rea-p-1 e Rea-w-1, e as instâncias aleatórias RND-9, RND-15, RND-18, RND-31, RND-39 e RND-45 com limites de 50, 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 segundos de CPU. Todos os códigos foram implementados em Java.

O tamanho da população inicial foi configurado em 40. Pré-testes indicaram que resultados não eram significativamente afetados para populações com tamanhos acima de 40 soluções. Tais resultados foram obtidos com base em 10 execuções independentes do algoritmo sobre três instâncias aleatoriamente selecionadas (Rea-a-1, Rea-w-4, Rnd-49) com um limite de tempo máximo de execução configurado em 50 segundos de CPU. A Figura 4 demonstra o desempenho médio do algoritmo em função do tamanho populacional para as instâncias citadas. O parâmetro Δ foi definido como 0,10, pois pré-testes indicaram que valores menores retornavam frequentemente soluções não-factíveis, e valores maiores desestimulavam o algoritmo a buscar soluções mais diversificadas.

FIGURA 4 – DESEMPENHO MÉDIO DO ALTP-GWO EM FUNÇÃO DO TAMANHO POPULACIONAL. AS BARRAS INDICAM A FUNÇÃO OBJETIVA MÉDIA, COMPRIMIDA NO INTERVALO [0,1] PARA MELHOR VISUALIZAÇÃO, DENTRE AS TRÊS INSTÂNCIAS SELECIONADAS.



FONTE: O autor (2020).

Os resultados obtidos pelo ALTP-GWO são comparados diretamente com os resultados obtidos pela heurística H-Poly, desenvolvida por Homsi et al. (2019). Os autores forneceram gentilmente os resultados detalhados do trabalho para comparação. Mesmo embora a configuração da CPU utilizada em Homsi et al. (2019) seja superior à CPU utilizada no presente trabalho, nenhuma forma de escalonamento foi utilizada nos resultados. A título de comparação, os autores implementaram seus códigos em C++, e executaram seus experimentos em um Intel Core i7-3960X 3.30 GHz com 64GB de memória RAM.

5.3 RESULTADOS EXPERIMENTAIS

Os resultados sobre as instâncias realísticas e aleatórias são reportados nas Tabelas 3 e 4, respectivamente. As duas primeiras colunas apresentam os resultados do *Gap* médio obtidos pelo H-Poly e o ALTP-GWO, respectivamente. As colunas remanescentes representam resultados do ALTP-GWO e apresentam o melhor e pior *Gap*, o número de itens transportados como percentagem das demandas *upper bound* da unidade, e o número de veículos utilizados relativo ao número total de veículos disponíveis para transporte, para cada instância. Os resultados em negrito indicam o melhor Gap encontrado pelos algoritmos em cada instância. Os experimentos são realizados para um limite de tempo de 50 segundos de CPU em ambos os algoritmos.

No que concerne às instâncias realísticas, o *Gap* médio obtido pelo ALTP-GWO sobre todas as instâncias analisadas foi 3,86%, enquanto que o *Gap* médio com H-Poly foi de 4,26%. A média dos melhores e piores *Gap*'s foram 2,68% e 6,77%, respectivamente. Em média, as soluções entregam 88,03% do *upper bound* do total de itens requeridos pela unidade, com a utilização de 1,63% da frota de veículos disponíveis para o transporte.

TABELA 3 – RESULTADOS SOBRE AS INSTÂNCIAS REALÍSTICAS COM BASE EM 30 EXECUÇÕES INDEPENDENTES PARA CADA CONJUNTO DE DADOS. NEGRITO REPRESENTA O MELHOR *GAP* ENTRE OS ALGORITMOS.

Instância	H-Poly	ALTP-GWO	Melhor *Gap*	Pior *Gap*	Itens entregues (% soma de *UB*)	Veículos utilizados (% do total disponível)
REA-a-1	4.26	**4.11**	2.68	6.11	87.31	1.61
REA-a-2	4.64	**4.55**	2.59	6.98	86.78	1.63
REA-a-3	4.99	**3.92**	2.10	5.33	86.84	1.60
REA-a-4	5.00	**3.67**	2.21	5.36	87.11	1.63
REA-a-5	**3.38**	3.87	2.72	5.26	87.65	1.61
REA-b-1	**2.66**	3.59	2.37	4.97	84.46	1.62
REA-b-2	**3.32**	4.78	2.73	7.87	85.96	1.61
REA-b-3	4.87	**4.34**	2.60	8.12	89.30	1.65
REA-b-4	**3.60**	3.82	2.47	6.04	88.27	1.64
REA-b-5	6.76	**4.51**	2.34	9.57	88.19	1.64
REA-c-1	**4.09**	4.48	3.05	7.11	88.47	1.64
REA-c-2	4.71	**4.41**	3.09	8.15	88.56	1.65
REA-c-3	**3.35**	4.30	2.23	6.31	88.57	1.63
REA-c-4	5.36	**4.21**	2.28	6.35	88.36	1.63
REA-c-5	5.10	**4.44**	2.63	7.10	88.33	1.63
REA-p-1	5.17	**3.55**	2.24	4.77	88.55	1.64
REA-p-2	4.37	**3.83**	2.00	6.72	88.33	1.64
REA-p-3	5.33	**3.63**	2.35	6.91	88.53	1.65
REA-p-4	4.11	**3.79**	2.41	9.07	88.65	1.65
REA-p-5	**3.51**	3.75	2.36	5.29	88.70	1.65
REA-q-1	3.34	**3.27**	2.43	7.57	88.46	1.34
REA-q-2	3.07	**2.85**	2.01	6.31	88.62	0.88
REA-q-3	3.14	**2.87**	2.27	4.90	88.25	0.63
REA-q-4	**2.34**	2.79	1.78	8.39	88.08	1.63
REA-q-5	3.41	**2.92**	2.24	7.97	88.63	1.21
REA-w-1	4.69	**3.83**	1.63	6.14	88.26	1.16
REA-w-2	3.84	**3.80**	2.09	5.40	88.45	0.83
REA-w-3	4.89	**4.24**	2.51	8.51	88.55	1.48

REA-w-4	5.98	**3.96**	2.41	8.41	88.29	1.30
REA-w-5	4.45	**3.79**	2.60	6.10	88.32	0.79
Média	4.26	**3.86**	2.38	6.77	88.03	1.63

FONTE: O autor (2020)

Por outro lado, o ALTP-GWO obteve um desempenho inferior ao H-Poly nas instâncias aleatórias. O *Gap* médio foi de 7,60% e 5,16% para o ALTP-GWO e H-Poly, respectivamente. Em média, as soluções obtidas pelo ALTP-GWO cobrem 85% do *upper bound* do total de demandas realizadas pela unidade, com uma utilização média de 2% da frota disponível. A média dos melhores e piores *Gap*'s foi de 6,68% e 8,50%, respectivamente.

TABELA 4 – RESULTADOS SOBRE AS INSTÂNCIAS ALEATÓRIAS COM BASE EM 30 EXECUÇÕES INDEPENDENTES PARA CADA CONJUNTO DE DADOS. NEGRITO REPRESENTA O MELHOR *GAP* ENTRE OS ALGORITMOS.

Instância	H-Poly	ALTP-GWO	Melhor *Gap*	Pior *Gap*	Itens entregues (% soma de *UB*)	Veículos utilizados (% do total disponível)
RND-1	**8.66**	12.50	11.61	13.40	83.56	2.96
RND-2	**4.63**	8.35	7.22	9.48	82.47	2.39
RND-3	**3.65**	9.49	8.48	10.50	78.78	1.94
RND-4	9.32	**8.63**	7.70	9.56	85.13	2.14
RND-5	**6.36**	10.71	9.10	12.32	77.62	1.76
RND-6	**4.85**	10.71	10.10	11.34	82.91	2.47
RND-7	**3.06**	4.75	4.00	5.49	84.44	1.33
RND-8	**5.21**	7.67	6.88	8.45	87.88	2.04
RND-9	**2.68**	5.02	2.92	5.23	85.73	1.49
RND-10	10.95	**10.64**	9.49	11.79	77.60	2.24
RND-11	9.65	**8.85**	8.23	9.46	87.78	2.29
RND-12	**9.77**	13.75	12.25	15.25	85.94	2.78
RND-13	**1.73**	5.03	4.34	5.72	91.23	1.80
RND-14	**3.36**	6.89	5.94	7.84	85.39	1.78
RND-15	**2.53**	5.89	5.15	6.63	94.38	1.92
RND-16	5.90	**5.84**	5.37	6.31	86.09	1.74
RND-17	6.58	**5.03**	3.60	6.45	89.73	1.93
RND-18	**4.49**	6.65	5.63	7.67	85.26	1.89
RND-19	**4.52**	6.36	5.08	7.65	84.23	2.02
RND-20	**2.76**	4.42	3.82	5.02	83.12	1.58
RND-21	**2.09**	8.81	7.33	10.28	81.75	1.60
RND-22	**3.41**	9.44	8.57	10.31	84.04	2.43
RND-23	**5.04**	10.67	10.01	11.32	79.02	2.36
RND-24	3.52	**3.16**	2.81	3.52	81.52	1.75
RND-25	**6.13**	9.69	7.93	11.45	91.51	1.85
RND-26	**5.40**	8.17	7.39	8.96	83.45	2.21
RND-27	**7.44**	9.94	7.87	10.02	87.04	2.22
RND-28	**2.05**	4.33	3.76	4.90	87.86	1.85
RND-29	**5.12**	8.14	7.24	9.03	86.10	2.02
RND-30	**10.07**	10.56	9.78	11.34	83.68	2.57
RND-31	**4.07**	4.99	4.38	5.60	83.23	1.92
RND-32	**3.25**	5.51	4.59	6.43	93.28	1.78
RND-33	15.52	**13.77**	12.22	15.31	79.31	2.42
RND-34	**6.33**	7.12	6.52	7.72	81.79	2.41
RND-35	**2.43**	5.69	4.73	6.64	90.75	1.52
RND-36	**2.45**	2.44	2.20	2.71	91.52	1.52
RND-37	**4.24**	4.75	4.45	5.05	83.30	1.97
RND-38	**5.93**	8.17	6.95	9.39	88.99	2.01
RND-39	**5.28**	8.67	7.89	9.45	82.95	1.89
RND-40	**4.07**	6.71	5.72	7.70	89.52	1.79
RND-41	**3.96**	6.13	5.70	6.56	81.86	2.01

RND-42	9.80	**8.45**	7.47	9.44	84.63	2.32
RND-43	**2.91**	7.11	5.60	8.63	86.32	1.80
RND-44	**3.96**	7.97	7.24	8.70	87.09	2.13
RND-45	**3.39**	6.59	5.95	7.24	80.13	1.92
RND-46	**2.88**	5.93	4.17	7.70	89.27	1.70
RND-47	**4.76**	6.08	5.64	6.53	89.85	2.00
RND-48	**6.08**	8.85	8.08	9.62	79.39	2.03
RND-49	**3.35**	11.24	10.16	12.33	74.60	1.92
RND-50	**2.57**	5.31	4.88	5.73	87.02	1.71
Média	**5.16**	7.60	6.68	8.50	85.00	2.00

FONTE: O autor (2020)

Resultados por limite de tempo nas instâncias realísticas (Tabela 5) indicam que o ALTP-GWO possui um desempenho superior ao H-Poly, em média, em todos os limites considerados. A melhoria no *Gap* para $\mathrm{TL} = 1000$ relativo ao $\mathrm{TL} = 50$ foi 37,4% para ALTP-GWO e 35,4% for H-Poly.

Os resultados nas instâncias aleatórias, por outro lado, indicam que o H-Poly possui um melhor desempenho dentro de limites de tempo mais restritos. Entretanto, como apresentado na Tabela 6, o H-Poly converge mais rapidamente que o ALTP-GWO e perde sua capacidade de busca após convergência. Esse fato é bastante claro na coluna para os *Gap*'s médios em ambas as Tabelas 5 e 6. A melhoria do *Gap*, nas instâncias aleatórias, em $\mathrm{TL} = 1000$ relativo ao $\mathrm{TL} = 50$ é 61.6% e 41.0% para ALTP-GWO e H-Poly, respectivamente.

TABELA 5 – RESULTADOS NAS SEIS INSTÂNCIAS REALÍSTICAS SELECIONADAS. RESULTADOS SÃO APRESENTADOS COM BASE EM 30 EXECUÇÕES INDEPENDENTES DO ALGORITMO ARA CADA CONJUNTO DE DADOS E LIMITE DE TEMPO. NEGRITO REPRESENTA O MELHOR *GAP* ENTRE OS ALGORITMOS.

TL	Rea-a-1		Rea-b-1		Rea-c-1		Rea-p-1		Rea-q-1		Rea-w-1		Average	
	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO
50	4,26	4,11	2,66	3,59	4,09	4,48	5,17	3,55	3,34	3,27	4,69	3,83	4,04	**3,26**
100	3,62	3,85	2,48	3,04	2,80	3,83	3,30	3,23	2,00	3,21	4,69	3,18	3,15	**2,91**
200	3,44	3,70	2,45	2,78	2,80	3,48	2,53	3,11	2,00	3,23	4,44	3,11	2,94	**2,77**
300	3,44	3,45	1,97	2,81	2,80	3,26	2,53	2,68	2,00	2,85	3,95	3,10	2,78	**2,59**
400	3,44	3,29	1,97	2,93	2,80	3,43	2,53	2,50	2,00	2,79	3,95	2,69	2,78	**2,52**
500	3,44	3,01	1,86	2,87	2,80	3,51	2,18	2,45	2,00	2,61	3,35	2,75	2,61	**2,46**
600	3,44	2,94	1,86	2,31	2,80	2,94	2,18	2,46	2,00	2,48	3,35	2,71	2,61	**2,26**
700	3,44	2,98	1,86	2,44	2,80	2,99	2,18	2,36	2,00	2,34	3,35	2,49	2,61	**2,23**
800	3,44	2,98	1,86	2,25	2,80	3,08	2,18	2,65	2,00	2,26	3,35	2,62	2,61	**2,26**
900	3,44	2,99	1,86	2,21	2,80	2,64	2,18	2,29	2,00	2,32	3,35	2,40	2,61	**2,12**
1000	3,44	2,83	1,86	2,01	2,80	2,66	2,18	2,42	2,00	2,16	3,35	2,23	2,61	**2,04**

TABELA 6 – RESULTADOS NAS SEIS INSTÂNCIAS ALEATÓRIAS SELECIONADAS. RESULTADOS SÃO APRESENTADOS COM BASE EM 30 EXECUÇÕES INDEPENDENTES DO ALGORITMO ARA CADA CONJUNTO DE DADOS E LIMITE DE TEMPO. NEGRITO REPRESENTA O MELHOR GAP ENTRE os algoritmos.

TL	RND-9		RND-15		RND-18		RND-31		RND-39		RND-45		Média	
	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO	H-Poly	GWO
50	2,68	5,02	2,53	5,89	4,48	6,65	4,07	4,99	5,28	8,67	3,40	6,59	**4,20**	6,30
100	2,22	4,82	2,53	5,25	2,09	5,70	3,74	4,46	4,76	7,59	2,98	6,04	**3,34**	5,64
200	2,22	3,31	1,81	4,57	1,96	5,26	2,95	3,97	3,45	7,96	2,41	4,84	**2,65**	4,99
300	2,22	3,09	1,81	3,87	1,96	4,98	2,71	4,32	3,09	6,00	2,56	4,42	**2,57**	4,45
400	2,22	2,53	1,81	3,88	1,96	4,87	2,71	3,40	3,05	5,17	2,05	4,32	**2,48**	4,03
500	2,22	2,90	1,81	3,68	1,96	5,50	2,71	3,56	3,05	5,65	2,05	3,97	**2,48**	4,21
600	2,22	2,25	1,81	3,69	1,96	4,85	2,71	3,33	3,05	4,91	2,05	3,71	**2,48**	3,79
700	2,22	2,61	1,81	3,50	1,96	5,00	2,71	3,46	3,05	4,85	2,05	4,01	**2,48**	3,91
800	2,22	1,76	1,81	3,46	1,96	4,92	2,71	3,19	3,05	4,86	2,05	3,39	**2,48**	3,60
900	2,22	1,76	1,81	3,35	1,96	3,51	2,71	2,98	3,05	4,01	2,05	2,76	**2,48**	3,06
1000	2,22	1,75	1,81	2,56	1,96	2,92	2,71	2,27	3,05	2,87	2,05	2,15	2,48	**2,42**

6 DISCUSSÃO

Resultados para ambas as instâncias, aleatórias e realísticas, indicam que, de uma perspectiva operacional, as soluções obtidas pelo ALTP-GWO são satisfatórias, uma vez que em ambos os grupos as soluções encontradas entregam em torno de 85% da soma do máximo de itens requeridos pela unidade. Ainda, os *Gap*'s médios permanecem dentro de 10% do melhor *upper bound* disponível, i.e., a solução do programa linear relaxado. Esse é o caso mesmo para as médias dos piores *Gap*'s, para as quais a média dos piores *Gap*'s é 6,77% para as instâncias realísticas, e 8,50% para as instâncias aleatórias.

Na Tabela 7 são reportadas as comparações do ALTP-GWO com o H-Poly (HOMSI et al., 2019) por grupo de instâncias, com $\mathrm{TL} = 50$. Nas instâncias realísticas o ALTP-GWO obteve, em média, resultados melhores, enquanto o H-Poly demonstra resultados superiores no grupo de instâncias aleatórias.

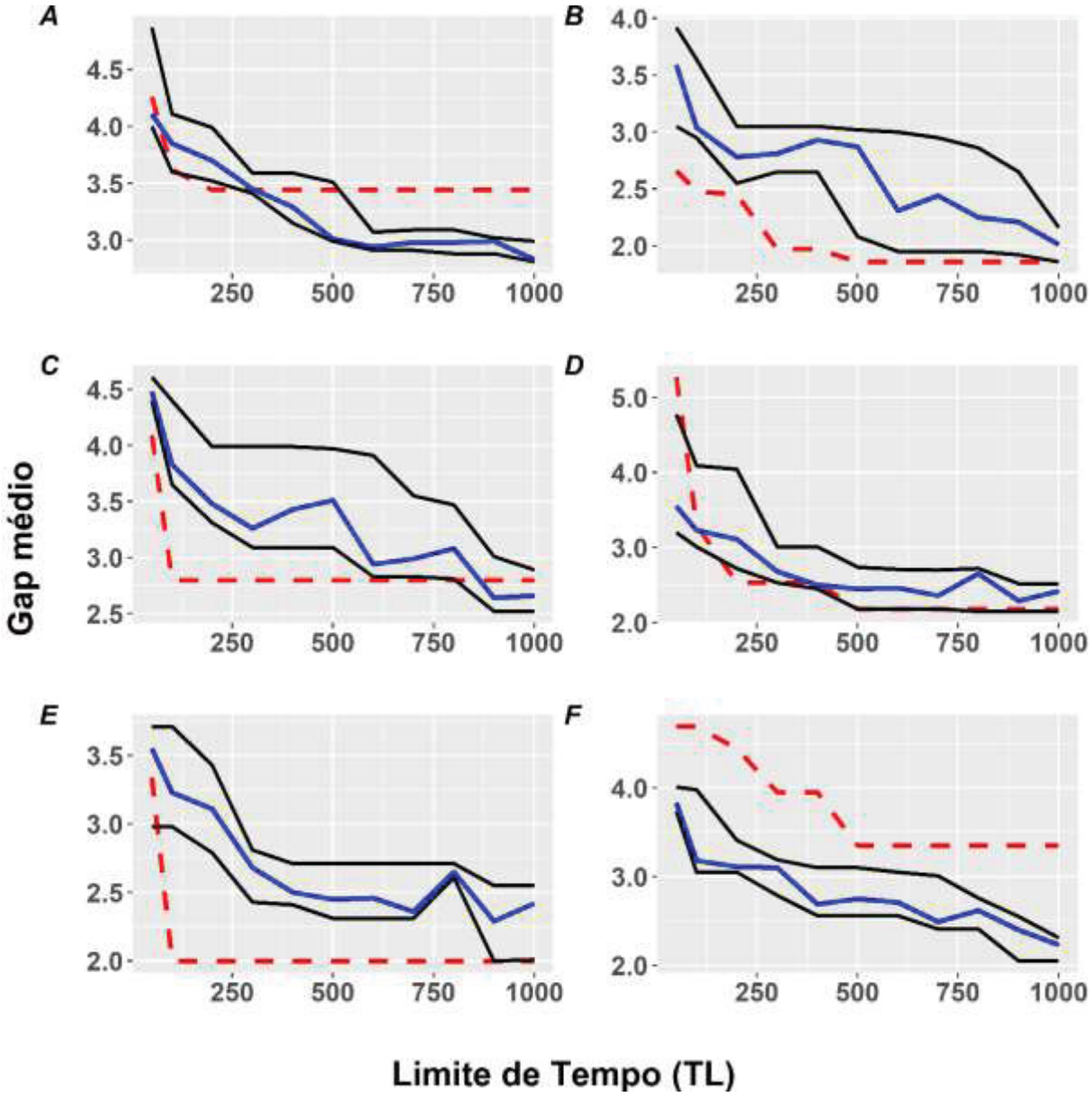
TABELA 7 – COMPARAÇÕES ENTRE O ALTP-GWO E OS *GAP*'S MÉDIOS OBTIDOS EM HOMSI ET AL. (2019) POR GRUPO DE INSTÂNCIAS. NEGRITO REPRESENTA O MELHOR *GAP* ENTRE ALGORITMOS.

Algoritmo	Rea-a	Rea-b	Rea-c	Rea-p	Rea-q	Rea-w	RND
H-Poly	4,45	4,24	4,52	4,50	3,06	4,77	**5,16**
ALTP-GWO	**4,02**	**4,21**	**4,37**	**3,71**	**2,94**	**3,92**	7,60

FONTE: O autor (2020)

O comportamento de ambos os algoritmos em diferentes limites de tempo, nas instâncias realísticas e aleatórias, é graficamente representado nas Figuras 5 e 6, respectivamente. Fica claro que H-Poly e o ALTP-GWO possuem diferentes comportamentos de busca em diferentes limites de tempo. Particularmente, o H-Poly converge muito mais rapidamente que o ALTP-GWO. Possivelmente essa característica decorre do fato de que a estrutura do H-Poly é construída com base em buscas locais que não sofrem grandes alterações ao longo da execução do algoritmo. Assim, o algoritmo rapidamente converge para ótimos locais e apresenta dificuldade encontrar soluções mais diversificadas.

FIGURA 5 – GAP MÉDIO EM FUNÇÃO DO LIMITE DE TEMPO (SEGUNDOS DE CPU). A LINHA TRACEJADA VERMELHA REPRESENTA H-POLY. A LINHA SÓLIDA AZUL REPRESENTA ALTP-GWO. AS LINHAS TRACEJADAS PRETAS REPRESENTAM OS MELHORES E PIORES GAPS ENCONTRADOS NAS 30 SIMULAÇÕES DO ALTP-GWO PARA CADA LIMITE DE TEMPO. A – REA-A-1; B – REA-B-1; C – REA-C-1; D – REA-P-1; E – REA-Q-1; F – REA-W-1.

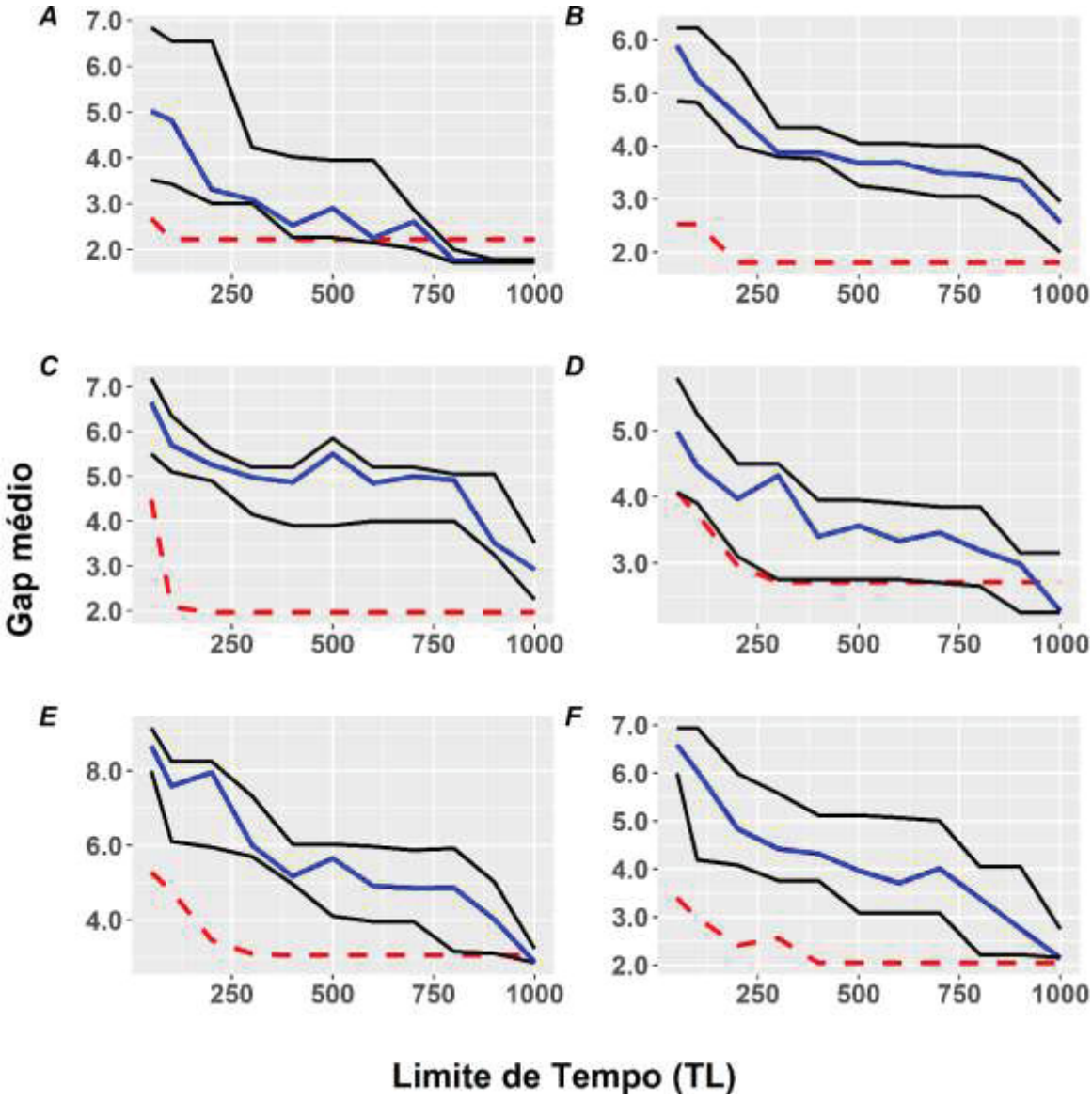


FONTE: O autor (2020)

O melhor desempenho do ALTP-GWO sobre todos os subgrupos de instâncias realísticas indica que o algoritmo não é sensível a variações nos parâmetros do problema individualmente. Entretanto, como as instâncias aleatórias não são subdivididas com as realísticas, o efeito combinado das variações nos parâmetros parece tornar o espaço de busca mais complexo e, assim, torna o ALTP-GWO menos

eficaz que o H-Poly (Tabela 6). Por outro lado, como os resultados sugerem, o ALTP-GWO possui uma capacidade de busca superior ao H-Poly em limites de tempo maiores. Assim, o algoritmo desenvolvido nesse trabalho pode ser eficientemente empregado em situações onde há maiores limites de tempo disponíveis para o cômputo das soluções. Também, é importante notar que um aumento em poder computacional pode afetar o desempenho do algoritmo.

FIGURA 6 – GAP MÉDIO EM FUNÇÃO DO LIMITE DE TEMPO (SEGUNDOS DE CPU). A LINHA TRACEJADA VERMELHA REPRESENTA O H-POLY. A LINHA SÓLIDA AZUL REPRESENTA O ALTP-GWO. AS LINHAS TRACEJADAS PRETAS REPRESENTAM OS MELHORES E PIORES GAPS ENCONTRADOS NAS 30 SIMULAÇÕES DO ALTP-GWO PARA CADA LIMITE DE TEMPO. A – RND-9; B – RND-15; C – RND-18; D – RND-31; E – RND-39; F – RND-45.



FONTE: O autor (2020)

Uma observação particularmente importante é a não estabilidade da população de soluções iniciais geradas pelo ALTP-GWO. Embora as Figuras 5 e 6 demonstrem um comportamento claramente convergente do algoritmo, i.e., produzindo soluções progressivamente melhores ao longo das iterações, algumas retrações na qualidade das soluções podem ser verificadas em limites de tempo maiores. Por exemplo, esse fenômeno pode ser claramente observado na Figura 5, gráfico C, em torno de TL=500. Provavelmente, a diversidade da população inicial de soluções possui tal magnitude que pode tornar as buscas pelo ALTP-GWO ineficientes para alguns conjuntos específicos de soluções geradas inicialmente. Posteriores estudos podem reduzir a diversidade das soluções por meio da inserção de buscas locais, e analisar o comportamento subsequente do ALTP para diferentes intensidades de buscas realizadas na população inicial.

Embora existam variações nos custos de designação e transporte, nas capacidades das bases e veículos, nas demandas das unidades, assim como no lucro de cada item, todas as instâncias permanecem homogêneas com respeito ao número de veículos disponíveis, o número de bases, e o número de tipos de itens a serem entregues para a unidade. Especificamente, todas as soluções utilizam no máximo 2% dos veículos disponíveis e, portanto, as variações em custos e lucros, causadas pelas diferenças na quantidade de veículos disponível, não parecem afetar profundamente a estrutura do problema. Assim, permanece desconhecido como ambos os algoritmos se comportariam no caso em que variações mais profundas fossem realizadas na estrutura das instâncias. De fato, como o problema possui potencial aplicação em situações de emergência, o comportamento do algoritmo deve ser testado em ambientes mais flexíveis, que permitirão identificar a capacidade de busca do ALTP-GWO em níveis mais críticos.

Testes do ALTP-GWO em situações em que diferentes bases e tipos de itens estão presentes também devem indicar com maior exatidão o comportamento do algoritmo. Variações nesses últimos parâmetros também podem ter forte influência sobre a estrutura do problema e, portanto, podem fornecer um ambiente mais complexo que permita verificar o comportamento do ALTP-GWO em uma variedade de situações distintas. Pesquisas posteriores podem ser orientadas à geração de novos conjuntos de instâncias com diferentes estruturas, assim como a construção de outros algoritmos evolutivos que permitam avaliar a eficiência de diferentes metaheurísticas para o ALTP.

7 CONSIDERAÇÕES FINAIS

O presente trabalho apresenta uma nova aplicação de uma variante recente do GWO. O problema sob consideração foi resolvido por meio da proposta de estratégias de atualização das posições para variáveis binárias e inteiras. Também, um gerador da população inicial e um reparador estocástico de soluções foi apresentado. O novo algoritmo foi comparado com a única heurística até então disponível na literatura, e provou possuir superioridade em diversas instâncias. Particularmente, as instâncias realísticas foram resolvidas com maior eficácia dentro de limites de tempo pequenos impostos ao algoritmo. Adicionalmente, resultados experimentais indicam que o novo algoritmo possui uma capacidade de busca mais flexível, em relação ao H-Poly, em limites de tempo maiores, o que pode ser uma vantagem quando situações em que limites de tempo menos restritos podem ser considerados. No entanto, instabilidades na qualidade das soluções ao longo de iterações com limites de tempo superiores sugerem a necessidade de reavaliação da estrutura do gerador da população inicial de soluções, o qual poderá ser combinado com, por exemplo, buscas locais que adicionem restrições à diversidade da população inicial e, assim, garantam a estabilidade do ALTP-GWO em execuções com limites de tempo maiores.

REFERÊNCIAS

ABDEL-BASSET, M.; EL-SHAHAT, D.; EL-HENAWY, I.; DE ALBUQUERQUE, V. H. C.; MIRJALILI, S. A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. **Expert Systems with Applications**, v. 139, p. 112824, 2020.

BIAN, X. Q.; ZHANG, Q.; ZHANG, L.; CHEN, J. A grey wolf optimizer-based support vector machine for the solubility of aromatic compounds in supercritical carbon dioxide. **Chemical Engineering Research and Design**, v. 123, p. 284-294, 2017.

DEWANGAN, R. K.; SHUKLA, A.; GODFREY, W. W. Three-dimensional path planning using Grey wolf optimizer for UAVs. **Applied Intelligence**, v. 49, n. 6, p. 2201-2217, 2019.

DIMITROV, N. B.; SOLOW, D.; SZMEREKOVSKY, J.; GUO, J. Emergency relocation of items using single trips: Special cases of the Multiple Knapsack Assignment Problem. **European Journal of Operations Research**, v. 258, n. 3, p. 938-942, 2017.

FARIS, H.; MIRJALILI, S.; ALJARAH, I. Automatic selection of hidden neurons and weights in neural networks using grey wolf optimizer based on a hybrid encoding scheme. **International Journal of Machine Learning and Cybernetics**, v. 10, n. 10, p. 2901-2920, 2019.

FARIS, H.; ALJARAH, I.; AL-BETAR, M. A.; MIRJALILI, S. Grey wolf optimizer: a review of recent variants and applications. **Neural Computing Applications**, v. 30, n. 2, p. 413-435, 2018.

GENDREAU, M.; POTVIN, J. Y. **Handbook of metaheuristics**. New York: Springer. 2010

GUPTA, S.; DEEP, K. (2019). An efficient grey wolf optimizer with opposition-based learning and chaotic local search for integer and mixed-integer optimization problems. **Arabian Journal for Science and Engineering**, v. 44, n. 8, p. 7277-7296, 2019.

HOMSI, G.; JORDAN, J.; MARTELLO, S.; MONACI, M. The assignment and loading transportation problem. **European Journal of Operations Research**, v. 289, n. 3, p. 999-1007, 2019.

HE, Y.; ZHANG, X.; LI, W.; WANG, J.; LI, N. An efficient binary differential evolution algorithm for the multidimensional knapsack problem. **Engineering with Computers**, 2019.

KAPOOR, S.; ZEYA, I.; SINGHAL, C.; NANDA, S. J. A grey wolf optimizer based automatic clustering algorithm for satellite image segmentation. **Procedia computer science**, v. 115, p. 415-422, 2017.

KATAOKA, S.; YAMADA, T. Upper and lower bounding procedures for the multiple knapsack assignment problem. **European Journal of Operations Research**, v. 237, n. 2, p. 440-447, 2014.

KOMAKI, G. M.; KAYVANFAR, V. Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. **Journal of Computational Science**, v. 8, p. 109-120, 2015.

KORAYEM, L.; KHORSID, M.; KASSEM, S. S. Using grey wolf algorithm to solve the capacitated vehicle routing problem. In IOP CONFERENCE SERIES: MATERIALS SCIENCE AND ENGINEERING, 2015. **Anais**…IOP Publishing, 2015, p. 012014.

LAYEB, A. A hybrid quantum inspired harmony search algorithm for 0–1 optimization problems. **Journal of Computational and Applied Mathematics**, v. 253, p. 14-25, 2013.

LUO, K.; ZHAO, Q. A binary grey wolf optimizer for the multidimensional knapsack problem. **Applied Soft Computing**, v. 83, p. 105645, 2019.

LUO, K. Enhanced grey wolf optimizer with a model for dynamically estimating the location of the prey. **Applied Soft Computing**, v. 77, p. 225-235, 2019.

LINET, O.; EDIZ, E.; BESTE, K. Emergency logistics planning in natural disasters. **Annals of Operations Research**, v. 129, n. 1-4, p. 217-245, 2004.

MARTELLO, S.; TOTH, P. **Knapsack problems**: Algorithms and computer implementations. Chichester: John Wiley & Sons Inc. (1990).

MARTELLO, S.; MONACI, M. Algorithmic approaches to the multiple knapsack assignment problem. **Omega**, v. 90, p. 102004, 2020.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey wolf optimizer. **Advances in Engineering Software**, v, 69, p. 46-61, 2014.

MIRJALILI, S. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. **Applied Intelligence**, v. 43, n. 1, p. 150-161, 2015.

MIRJALILI, S.; ALJARAH, I.; MAFARJA, M.; HEIDARI, A. A.; FARIS, H. Grey Wolf optimizer: theory, literature review, and application in computational fluid dynamics problems. In **Nature-inspired optimizers. Springer, Cham**. p. 87-105, 2020.
.
OZSOYDAN, F. B. Effects of dominant wolves in grey wolf optimization algorithm. **Applied Soft Computing**, v. 83, p. 105658, 2019.

PUCHINGER, J.; RAIDL, G. R.; PFERSCHY, U. The multidimensional knapsack problem: Structure and algorithms. **INFORMS Journal on Computing**, v. 22, n. 2, p. 250-265, 2010.

QU, C.; GAI, W.; ZHONG, M.; ZHANG, J. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning. **Applied Soft Computing**, v. 89, p. 106099, 2020.

RASHID, T. A.; ABBAS, D. K.; TUREL, Y. K. A multi hidden recurrent neural network with a modified grey wolf optimizer. **PloS one**, v. 14, n. 3, e0213237, 2019.

ROOPA DEVI, E. M.; SUGANTHE, R. C. Enhanced transductive support vector machine classification with grey wolf optimizer cuckoo search optimization for intrusion detection system. **Concurrency and Computation: Practice and Experience**, v. 32, n. 4, e4999, 2020.

SAXENA, A.; SHEKHAWAT, S. Ambient air quality classification by grey wolf optimizer-based support vector machine. **Journal of environmental and public health**, v. 2017, 2017.

TRIPATHI, A. K.; SHARMA, K.; BALA, M. A novel clustering method using enhanced grey wolf optimizer and MapReduce. **Big data research**, v. 14, p. 93-100, 2018.

WANG, R.; XIA, K.; SANDRINE, M. Grey Wolf Optimizer with Multi-Strategy Optimization and Its Application on TSP. In PROCEEDINGS OF THE 2020 INTERNATIONAL CONFERENCE ON INTERNET COMPUTING FOR SCIENCE AND ENGINEERING, 1996. **Anais**… ACM-Association for Computing Machinery, 2020, p. 52-58.

WANG, L.; ZHENG, X. L.; WANG, S. Y. A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. **Knowledge-Based Systems**, v. 48, 17-23, 2013

ZHENG, Y. J.; LING, H. F.; XU, X. L.; CHEN, S. Y. Emergency scheduling of engineering rescue tasks in disaster relief operations and its application in China. **International Transactions in Operational Research**, v. 22, n. 3, p. 503-518, 2015.

ZHENG, Y. J.; CHEN, S. Y.; LING, H. F. Evolutionary optimization for disaster relief operations: A survey. **Applied Soft Computing**, v. 27, p. 553-566, 2015.

ZHANG, S.; ZHOU, Y.; LI, Z.; PAN, W. Grey wolf optimizer for unmanned combat aerial vehicle path planning. **Advances in Engineering Software**, v. 99, p. 121-136, 2016.

ZHANG, B.; PAN, Q. K.; ZHANG, X. L.; DUAN, P. Y. An effective hybrid harmony search-based algorithm for solving multidimensional knapsack problems. **Applied Soft Computing**, v. 29, p. 288-297, 2015.